高等教育“十三五”规划教材

数据库原理及应用

主　审　耿红琴

主　编　张银玲　杨锋英

副主编　葛文庚　徐　亮　张得生

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

VISIT…

内 容 简 介

本书以应用为目的，以能力培养为目标，以 JSP 和 SQL Server 2008 为平台，以案例为引导，主要介绍数据库的基本概念，数据模型，关系代数，关系数据理论，数据库设计，数据库完整性约束，SQL Server 2008、JDK、Tomcat、Eclipse 的安装和配置，基于 JSP 的学生成绩管理系统的设计与实现，SQL 查询和数据操纵，存储过程，触发器，游标，数据库的备份和恢复，数据库的安全控制，基于 MVC 设计模式的学生成绩管理系统的设计与实现，SQL 注入，数据加密等。每章由学习目标、主要内容、重点与难点、教学案例、相关知识、习题和实训项目等构成。

本书以“学生成绩管理系统”为案例，通过案例，将数据库的相关知识恰当地融入到案例的分析和设计过程中，图文并茂、深入浅出、通俗易懂、强调实践、突出应用，使读者在学习过程中不仅能掌握数据库的相关知识，而且能培养其综合分析问题和解决问题的能力。全书采用案例方式安排教学内容，注重实用性和可操作性，有助于提高实际动手能力。

本书可作为应用型和技能型人才培养的高等学校和高职高专院校计算机及相关专业数据库方面课程教材，也可作为计算机培训机构的数据库培训教材，还可供数据库软件应用和开发人员学习参考使用。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

数据库原理及应用 / 张银玲，杨锋英主编. —北京：电子工业出版社，2016.7
ISBN 978-7-121-28969-9

Ⅰ. ①数… Ⅱ. ①张… ②杨… Ⅲ. ①数据库系统Ⅳ. ①TP311.13

中国版本图书馆 CIP 数据核字（2016）第 124607 号

策划编辑： 祁玉芹
责任编辑： 张瑞喜
印　　刷： 中国电影出版社印刷厂
装　　订： 中国电影出版社印刷厂
出版发行： 电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本： 787×1092　1/16　印张：22.5　字数：548 千字
版　　次： 2016 年 7 月第 1 版
印　　次： 2016 年 7 月第 1 次印刷
定　　价： 48.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前言

PREFACE

数据库技术是信息系统的核心和基础，是信息化建设的基础设施，数据库信息量的大小和使用频率已成为衡量一个国家信息化程度的重要标志，因此，作为当代的大学生，学习和掌握数据库知识是非常必要的，作为计算机相关专业的学生更要学好数据库技术。

“数据库原理及应用”是计算机类专业的核心课程，也是信息处理类专业的必修课程。课程教学目标是系统地介绍数据库基本原理，并结合具体的学生成绩管理系统介绍数据库理论知识的具体应用。通过该课程的学习，使读者在数据库应用软件的开发过程中能够设计合理的数据库，从而提高软件开发的质量，同时能够进行数据库的查找和维护操作，达到熟练地应用数据库技术的目的。

数据库基本原理抽象又很重要，但是，其实践操作更为重要。目前很多教材数据库的基本原理知识介绍非常详细，但是实践部分内容不足，或前后章节案例独立，学生获得的对完整系统后台数据库的设计和维护知识不完整。鉴于此，编者结合自己多年来的数据库理论及数据库技术实践教学经验编写了本教材。

本书的特点是，以应用为目的，以能力培养为目标，案例引导，内容全面，通俗易懂，图文并茂，知识连贯，概念清晰，实践性强。全书所有案例围绕一个完整的学生成绩管理系统展开，将数据库的相关知识恰当地融入到案例的分析和设计过程中，通过一个案例体现其应用。使学生通过案例的学习，掌握数据库知识及其在实际项目中的使用，避免学生学习了数据库理论而不知道如何使用的现象。同时，通过案例学习使读者对数据库的设计有一个完整、清晰的思路。书中每个案例后面配有针对性的习题，部分任务后面还配有具体的实训项目，供读者练习，有助于增强读者对相关知识的理解和实际应用能力的培养。另外，本书还配有电子教学参考资料（书中所用数据库脚本、所有案例的源代码、电子课件、习题和实训项目参考答案）。

全书共分 4 个模块，模块 1 是数据库理论基础；模块 2 是开发环境的搭建与使用；模块 3 是学生成绩管理系统的设计与实现；模块 4 是学生成绩管理系统优化。本书建议教学学时为 64 学时，其中理论 32 学时，实验 32 学时。书中第 1 和第 3 模块为教学重点，其中第 3 模块为教学难点，应分配较多的学时。

本书由张银玲和杨锋英主编，葛文庚、徐亮和张得生为副主编，参加编写的还有李俊峰。书中模块 1 的任务 1、任务 2、任务 3 由徐亮编写；模块 1 的任务 4 由李俊峰编写；模块 1 任务 5 中的 1～6 和模块 2 的案例 1 由张得生编写;模块 1 任务 5 中的 7 和模块 2 的案例 2 由葛文庚编写；模块 3 由杨锋英编写；模块 4 由张银玲编写。全书由耿红琴审核并定稿。

本书既可以作为高等院校和高职院校的数据库原理及应用课程的教材，也可以作为有一定的面向对象编程基础和数据库基础进行 Web 应用程序开发人员的参考资料。

在编写本书的过程中参考了相关文献，在此向这些文献的作者深表感谢。同时，在此对参与了本书配套案例“学生成绩管理系统”的设计与开发工作的李鹏龙、刘洋同学及对内容和实验提出了许多中肯有益意见的马亚东、刘志东、王广智同学表示衷心感谢。由于时间较紧，书中难免有错误与不足之处，恳请专家和广大读者批评指正。

编　者

2016 年 4 月

目录

CONTENTS

模块 1　数据库理论基础……1

任务 1　数据库基础……1
习题……9
任务 2　关系数据库……13
习题……28
任务 3　关系数据理论……29
习题……34
任务 4　数据库设计……36
习题……55
任务 5　数据库的完整性约束机制……56
习题……65

模块 2　开发环境的搭建与使用……67

案例 1　安装并熟悉 SQL Server 2008……67
相关知识……82
案例 2　安装并熟悉 Java Web 开发环境……95
任务 1　JDK 的安装与配置……95
任务 2　Tomcat 的安装与配置……102
任务 3　安装 Eclipse 并熟悉 Eclipse 环境……109
相关知识……118
实训项目……121

模块 3　学生成绩管理系统的设计与实现 ……122

案例 1　学生成绩管理系统的设计 …… 122
任务 1　学生成绩管理系统的需求描述 …… 122
任务 2　学生成绩管理系统的概念结构设计 …… 123
任务 3　学生成绩管理系统的逻辑结构设计 …… 125
任务 4　学生成绩管理系统的物理结构设计 …… 126
任务 5　学生成绩管理系统的数据库实现 …… 126
任务 6　修改数据库中表结构 …… 129
习题 …… 130
实训项目 …… 130
相关知识 …… 130
案例 2　学生成绩管理系统的登录 …… 132
任务 1　设置系统编码格式 …… 132
任务 2　建立并运行 Web 项目 …… 134
习题 …… 147
相关知识 …… 147
习题 …… 158
拓展知识 …… 168
案例 3　教师功能的设计与实现 …… 173
任务 1　查询所授课程的成绩信息 …… 173
习题 …… 176
任务 2　其他类型查询 …… 176
习题 …… 187
任务 3　增加一个学生成绩 …… 188
习题 …… 192
相关知识 …… 192
任务 4　修改学生成绩 …… 194
习题 …… 200

相关知识 …… 200
任务 5　删除学生成绩 …… 201
习题 …… 203
相关知识 …… 203
案例 4　密码修改界面的设计与实现 …… 204
习题 …… 209
案例 5　学生成绩管理系统的备份与恢复 …… 209
任务 1　数据库备份 …… 210
习题 …… 221
任务 2　数据库恢复 …… 221
习题 …… 225
相关知识 …… 225
习题 …… 244
模块 4　学生成绩管理系统优化 …… 245
案例 1　开发 MVC 模式的学生成绩管理系统 …… 245
任务 1　创建不同模式对象的包 …… 246
任务 2　创建实体对象 …… 247
任务 3　创建数据处理模型 …… 251
任务 4　实现学生成绩管理系统的用户登录 …… 252
相关知识 …… 258
任务 5　教师功能的设计与实现 …… 263
相关知识 …… 279
任务 6　管理员功能的设计与实现 …… 314
相关知识 …… 319
习题 …… 324
实训项目 …… 324
案例 2　用户密码加密 …… 325
任务 1　添加注册页面 …… 325

任务 2　建立加密类 …… 326
任务 3　完成注册用户密码加密 …… 327
相关知识 …… 330
习题 …… 331
实训项目 …… 331
案例 3　成绩管理系统的安全控制 …… 331
任务 1　创建学生成绩管理系统数据库用户 …… 331
任务 2　创建学生成绩管理系统数据库学生角色 …… 340
任务 3　为学生成绩管理系统数据库用户和角色赋予权限 …… 345
相关知识 …… 349
习题 …… 352
实训项目 …… 352

模块 1　数据库理论基础

学习目标

通过本模块的学习了解数据库的基本概念及数据库管理技术的发展过程；掌握关系数据模型的设计过程；了解数据库系统结构及其组成；熟悉关系数据库的三个组成要素；掌握关系代数的相关运算；熟练掌握关系优劣的判定及优化；掌握数据库设计的基本步骤和数据库的完整性约束机制。

主要内容

（1） 数据库基础。
（2） 关系数据库。
（3） 关系数据理论。
（4） 数据库设计。
（5） 数据库的完整性约束机制。

重点与难点

（1） 关系代数。
（2） 关系数据理论。
（3） 概念结构设计。
（4） 逻辑结构设计。
（5） 触发器。

任务 1　数据库基础

一、四个基本概念

1. 数据（Data）

数据是数据库中存储的基本对象。由于早期的计算机常用于科学计算，因此所使用的数据多数是整数或者浮点数，因此通常人们对数据的理解是数值型数据是数据，但实质上数据类型有数值型数据、文本数据、图形、图像、视频、音频、学生的成绩信息、超市货

物销售信息等。所有的数据与其语义是密不可分的，如下一组数据：

（刘冰，2010，北京大学，计算机）

这组数据如果没有语义约定，则可以理解为：2010 年刘冰在北京大学学习计算机专业，或者理解为：刘冰在 2010 年离开北京大学去外地从事计算机生意，或者理解为：刘冰是 2010 年在北京大学从事计算机教育工作。

如果对这组数据加上相应的语义：

（姓名，学习时间，学习地点，学习专业）

则对该组数据只能有一种确定的理解，即 2010 年刘冰在北京大学学习计算机专业。因此，所有进入数据库中的数据必须有相应的语义解释。

2. 数据库（DataBase，简写为 DB）

顾名思义就是按照数据结构来组织、长期存储和管理数据的仓库。它产生于 60 年前，随着信息技术和市场的发展，特别是 20 世纪 90 年代以后，数据管理不再仅仅是存储和管理数据，而转变成用户所需要的各种数据的管理。数据库有很多种类型，从最简单的存储有各种数据的表格到能够进行海量数据存储的大型数据库系统都在各个方面得到了广泛的应用。数据库中数据的三个基本特点是长期存储、有组织、可共享。

3. 数据库管理系统（DataBase Management System，简称 DBMS）

数据库管理系统是一种操纵和管理数据库的大型系统软件，用于建立、使用和维护数据库，它对数据库进行统一的管理和控制，以保证数据库的安全性和完整性。用户通过 DBMS 访问数据库中的数据，数据库管理员也通过 DBMS 进行数据库的维护工作。常见的数据库管理系统软件有：Sybase、DB2、Oracle、SQL Server、Access、PostgreSQL、Informix 等。DBMS 提供的功能有以下几个方面：

（1） 数据定义：DBMS 提供数据定义语言 DDL（Data Definition Language），主要用于建立、修改数据库的库结构。DDL 所描述的库结构仅仅给出了数据库的框架，数据库的框架信息被存放在数据字典（Data Dictionary）中。

（2） 数据操纵：DBMS 提供数据操纵语言 DML（Data Manipulation Language），供用户实现对数据的增加、删除、修改、查询等操作。

（3） 数据库的运行管理：数据库的运行管理功能是 DBMS 的运行控制、管理功能，包括多用户环境下的并发控制、安全性检查和存取限制控制、完整性检查和执行、运行日志的组织管理、事务的管理和自动恢复，即保证事务的原子性。这些功能保证了数据库系统的正常运行。

（4） 数据组织、存储与管理：DBMS 要分类组织、存储和管理各种数据，包括数据字典、用户数据、存取路径等，需确定以何种文件结构和存取方式在存储级上组织这些数据，如何实现数据之间的联系。数据组织和存储的基本目标是提高存储空间利用率，选择合适的存取方法提高存取效率。

（5） 数据库的维护：这一部分包括数据库的数据载入、转换、转储，数据库的重组合重构以及性能监控等功能，这些功能分别由各个实用程序来完成。

数据库管理系统是数据库系统的核心，是管理数据库的软件。数据库管理系统就是实现把用户意义下抽象的逻辑数据处理，转换成为计算机中具体的物理数据处理的软件。有

了数据库管理系统，用户就可以在抽象意义下处理数据，而不必顾及这些数据在计算机中的布局和物理位置。

4. 数据库系统（DataBase System，简称 DBS）

数据库系统是指在计算机系统中引入数据库后的系统构成，一般由数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员和用户构成。应当指出的是，数据库的建立、使用和维护等工作只靠一个 DBMS 远远不够，还要有专门的人员来完成，这些人被称为数据库管理员（Data Base Administrator，简称 DBA）。

在一般不引起混淆的情况下常常把数据库系统简称为数据库。

二、数据管理技术

数据管理技术是指对数据进行分类、组织、编码、存储、检索和维护的技术，经历了人工管理、文件管理、数据库管理三个阶段。数据管理技术是应数据管理任务的需要而产生的，主要有数据处理和数据管理两方面内容，数据的处理是指对各种数据进行收集、存储、加工和传播的一系列活动的总和；数据管理则是指对数据进行分类、组织、编码、存储、检索和维护，它是数据处理的中心问题。

1. 人工管理时期

在计算机出现之前，人们运用常规的手段从事记录、存储和对数据加工，也就是利用纸张来记录和利用计算工具（算盘、计算尺）来进行计算，并主要使用人的大脑来管理和利用这些数据。

到了 20 世纪 50 年代中期，计算机主要用于科学计算。当时没有磁盘等直接存取设备，只有纸带、卡片、磁带等外存，也没有操作系统和管理数据的专门软件。数据处理的方式是批处理。该阶段管理数据的特点是：

（1） 数据不保存。因为当时计算机主要用于科学计算，对于数据保存的需求尚不迫切。

（2） 系统没有专用的软件对数据进行管理，每个应用程序都要包括数据的存储结构、存取方法和输入方法等。程序员编写应用程序时，还要安排数据的物理存储，因此程序员负担很重。

（3） 数据不共享。数据是面向程序的，一组数据只能对应一个程序。

（4） 数据不具有独立性。程序依赖于数据，如果数据的类型、格式或输入/输出方式等逻辑结构或物理结构发生变化，则必须对应用程序做出相应的修改。

2. 文件系统阶段

20 世纪 50 年代后期到 60 年代中期，随着计算机硬件和软件的发展，磁带、磁盘等直接存取设备开始普及，这一时期的数据处理系统是把计算机中的数据组织成相互独立的被命名的数据文件，并可按文件的名字来进行访问，对文件中的记录进行存取的数据管理技术。数据可以长期保存在计算机外存上，可以对数据进行反复处理，并支持文件的查询、修改、插入和删除等操作，这就是文件系统。文件系统数据管理示意如图 1-1 所示。

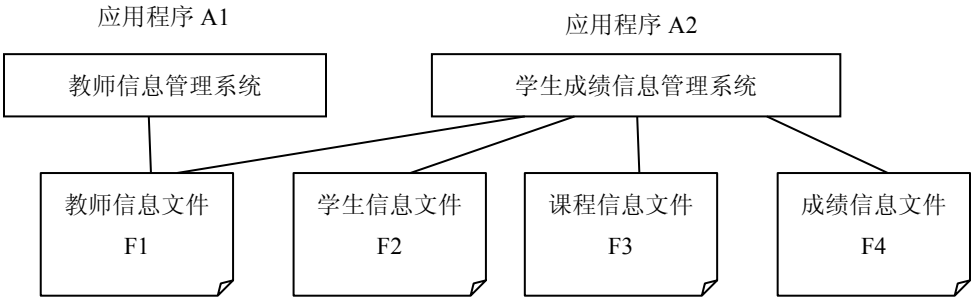


图 1-1　文件管理实现示例

文件系统实现了记录内的结构化，但从文件的整体来看却是无结构的。其数据面向特定的应用程序，因此数据共享性、独立性差，且冗余度大，管理和维护的代价也很大。

3. 数据库系统阶段

20 世纪 60 年代后期以来，计算机在软件和硬件方面都得到较大发展，更重要的是出现了大容量磁盘，存储容量大大增加且价格逐步下降，在此基础之上，基本克服了文件系统管理数据时的不足，并且满足和解决实际应用中多个用户、多个应用程序共享数据的要求，从而使数据能为尽可能多的应用程序服务，这就出现了数据库这样的数据管理技术。文件系统数据管理示意如图 1-2 所示。

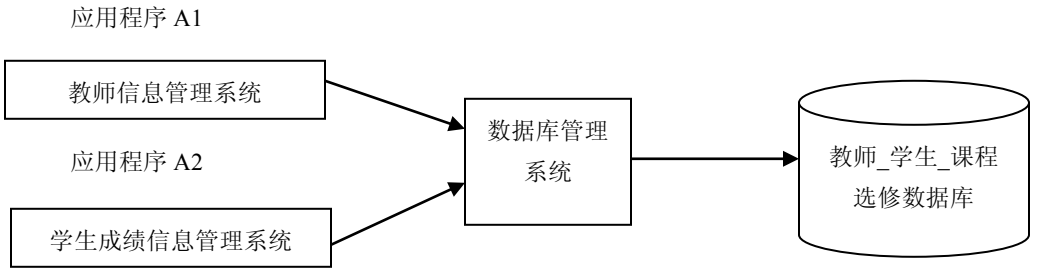


图 1-2　数据库管理实现示例

数据库的特点是数据不再只针对某一个特定的应用，而是面向全组织，具有整体的结构性，共享性高，冗余度减小，具有一定的程序与数据之间的独立性，并且对数据进行统一的控制。

此阶段的特点：

（1） 数据结构化。在描述数据时不仅要描述数据本身，还要描述数据之间的联系。数据结构化是数据库的主要特征之一，也是数据库系统与文件系统的本质区别。

（2） 数据共享性高、冗余少且易扩充。数据不再针对某一个应用，而是面向整个系统，数据可被多个用户和多个应用共享使用，而且容易增加新的应用，所以数据的共享性高且易扩充。数据共享可大大减少数据冗余。

（3） 数据独立性高。数据在磁盘上的数据库中怎样存储是由 DBMS 管理的，用户程序不需要了解，应用程序要处理的只是数据的逻辑结构，当数据的物理存储改变时，应用

程序不用改变；同时，数据的逻辑结构改变了，用户程序也可以不变。

（4） 数据由 DBMS 统一管理和控制。数据库为多个用户和应用程序所共享，对数据的存取往往是并发的，即多个用户可以同时存取数据库中的数据，甚至可以同时存放数据库中的同一个数据，为确保数据库数据的正确有效和数据库系统的有效运行，数据库管理系统提供以下 4 方面的数据控制功能。

① 数据安全性控制：防止因不合法使用数据而造成数据的泄露和破坏，保证数据的安全和机密。

② 数据的完整性控制：系统通过设置一些完整性规则，以确保数据的正确性、有效性和相容性。

③ 并发控制：多用户同时存取或修改数据库时，防止相互干扰而给用户提供不正确的数据，并使数据库受到破坏。

④ 数据恢复：当数据库被破坏或数据不可靠时，系统有能力将数据库从错误状态恢复到最近某一时刻的正确状态。

数据库技术的研究领域主要有三个领域：数据库管理系统软件的研制、数据库设计、数据库理论。

三、数据模型

我们把客观存在的事物以数据的形式存储到计算机中，经历了对现实生活中事物特性的认识、概念化到计算机数据库里的具体表示的逐级抽象过程，即现实世界－概念世界－机器世界三个领域。通常把概念世界称为信息世界；将机器世界称为存储或数据世界。

1. 三个世界

（1） 现实世界：我们管理的对象存于现实世界中。现实世界中事物与事物之间存在着联系，这种联系是客观存在的，是由事物本身的性质决定的。例如学校使用的教务系统中有教师、学生和课程三类对象，这些对象之间存在教师为学生授课、学生选修课程并取得成绩的联系。

（2） 概念世界：概念世界是我们头脑中对现实世界的客观反映，是对客观事物及其联系的一种抽象描述而产生的。概念模型是现实世界到机器世界必然经过的中间层次，由数据模型表示，当前流行的 DBMS 所支持的数据模型分为三种：层次模型、网状模型、关系模型，每种概念模型都涉及到下面几个术语：

① 实体：我们把客观存在并且可以相互区别的事物称为实体。实体可以是物理存在的事物，也可以是抽象事件，如学生、课程、学生选修课程后的成绩、教师等都是实体。

② 属性：描述实体或者联系的特性或者性质的数据项称为属性，属于一个实体所有实例都具有的性质。利用属性便于我们区分每一个实体 A 是 A，不是 B，如实体学生的学号、姓名、班级、性别、出生日期、家庭地址等是属性，我们通过学生的学号以区分每一个学生，像学生的学号这样，能够唯一标识实体的属性或最小属性组合，称为实体的码，又称为候选码或者关键字。如学生实体中的“学号”、教师实体中的“教工号”均可作为码。

③ 联系：实体之间的对应关系称为联系，它反映现实世界事物之间的相互关联。联系分为两种，一种是实体内部各属性之间的联系。另一种是实体之间的联系。例如一个学

校有多个班级；每个班级有一名班长；一个班级有多名学生组成；一门课程由多名学生选修，一名学生选修多门课程等都是实体内部或者实体与实体之间存在的联系，这种联系是客观存在的。

（3） 机器世界：存入计算机系统里的数据是将概念世界中的事物数据化的结果。为了准确地反映事物本身及事物之间的各种联系，数据库中的数据必须有一定的结构，这种结构用数据模型来表示。数据模型将概念世界中的实体，及实体间的联系进一步抽象成便于计算机处理的方式。

2. E-R 模型

E-R 模型（又称实体-联系模型）简称 E-R 图。它是描述概念世界，建立概念模型的实用工具。E-R 图包括三个要素：

实体——用矩形框表示，框内标注实体名称。

属性——用椭圆或者圆角矩形框表示，并在框内写上属性的名字，然后用连线与实体连接起来。

如学生实体有学号、姓名、班级、性别、出生日期、家庭住址属性，用 E-R 图表示如图 1-3 所示。

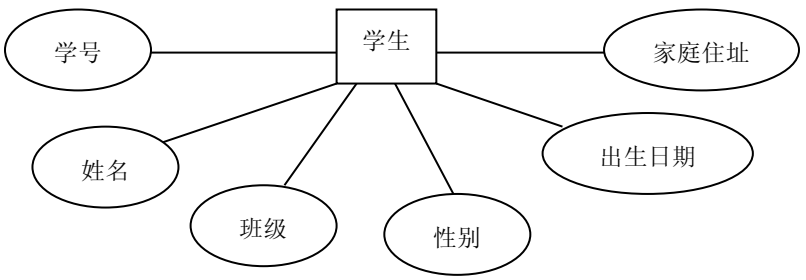


图 1-3　学生实体属性图

实体之间的联系——用菱形框表示，框内标注联系名称，用连线将菱形框分别与对应实体相连，并在连线上注明联系类型，如果两个实体发生联系时产生新属性，则该属性不属于两个实体中的任何一个，它属于联系，因此要用连线与该联系相连。如单独的课程是没有成绩属性的，而学生如果没有选修行为发生时，也没有成绩属性，当学生选修课程成功后，一定会产生成绩属性，因此该属性必然属于联系“选修”。

联系归结为三种类型：

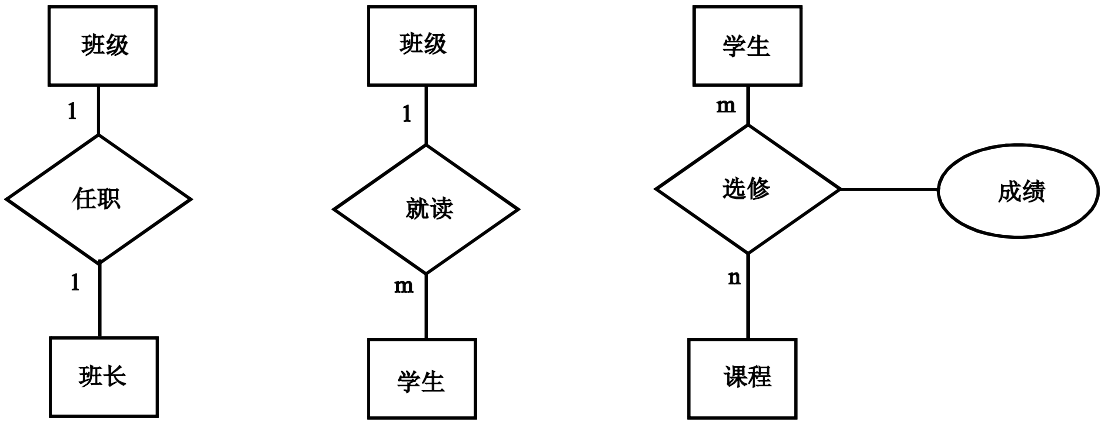
（1） 一对一的联系（1:1）。

设 A、B 为两个实体，若 A 中的每个实例至多和 B 中的一个实例有联系，反过来，B 中的每个实例至多和 A 中的一个实例有联系，称 A 与 B 是一对一的联系，记作 1:1。注意，1:1 联系不一定都是一一对应的关系。可能存在着无对应。如一个班级只有一个正班长，一个正班长不能同时在其他班级再兼任正班长，但对于新报到的学生班级，可能班长暂缺，故班级与班长之间属于一对一的联系，如图 1-4（a）所示。

（2） 一对多的联系（1:m）。

如果 A 实体中的每个实例可以和 B 中的几个实例有联系，而 B 中的每个实例与 A 中

的一个实例有联系，那么 A 对 B 属于一对多的联系，记作 1:m，如一个班级有多名学生，而一名学生只在一个班级就读，班级与学生之间属于一对多的联系，如图 1-4（b）所示。



（a）1:1　　（b）1:m　　（c）m:n

图 1-4　两个实体之间的三种联系实例

（3） 多对多的联系（m:n）。

若实体 A 中的每个实例可与实体 B 中的多个实例有联系，反过来，B 中的每个实体也可以与 A 中的多个实体有联系，称 A 对 B 或 B 对 A 是多对多的联系，记作 m:n，如一个学生可以选修多门课程，一门课程由多个学生选修，学生和课程间存在多对多的联系，并且当选修联系发生时，将产生新属性成绩，如图 1-4（c）所示。

E-R 图的设计过程是抽取实体、确定属性、定义联系、绘制 E-R 图四步组成。

例如，学生_选修系统的需求描述为：一个院系有一名辅导员，这名辅导员管理多个班级，每个班级有多名学生，并且不能在该校内出现双学位的情况；每个学生选修多门课程，每门课程也有多个学生选修。

首先，抽取实体。

鉴于实体一定是名词，但名词不一定是实体的原则，对上述需求的描述，从中找出名词：院系、辅导员、班级、学生、双学位、课程。分析名词发现，“双学位”仅是限制学生在该学校中只属于一个班级罢了，并不与其他产生联系，因此舍弃，其他两两之间均有联系，因此，保留得到实体有：院系、辅导员、班级、学生、课程。

其次，确定属性。

实际生活中，每个实体的属性可能有几十个甚至几百个不等，但对于不同的应用系统，只要把每个实体对于本系统应该需要的属性给出即可，如院系的属性通常有：院系编号、院系名称、院系联系电话、院系办公地点、院系负责人、院系成立日期、院系专业数、院系教师总人数、院系学生总人数等，对于本应用，仅需要将院系编号、院系名称、院系联系电话、院系办公地点四个属性用上即可。绘制每个实体的属性如图 1-5 至图 1-9 所示。

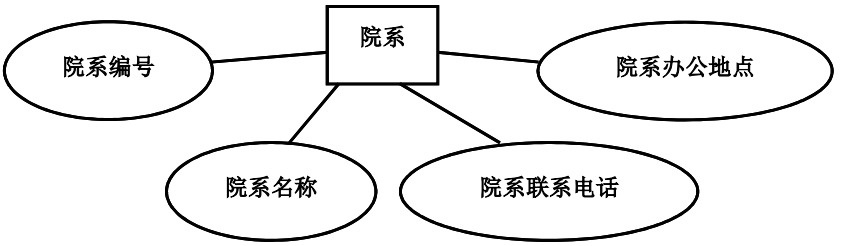


图 1-5　院系实体属性图

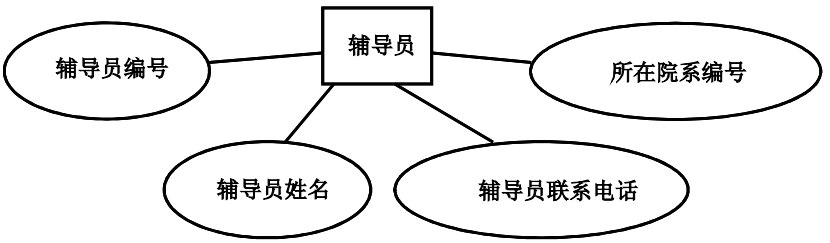


图 1-6　辅导员实体属性图

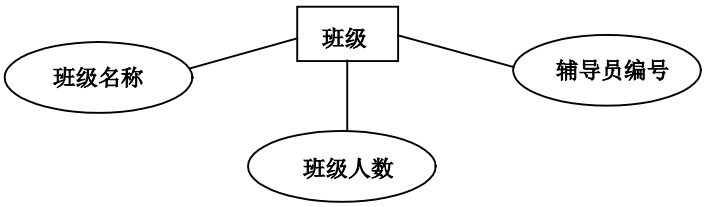


图 1-7　班级实体属性图

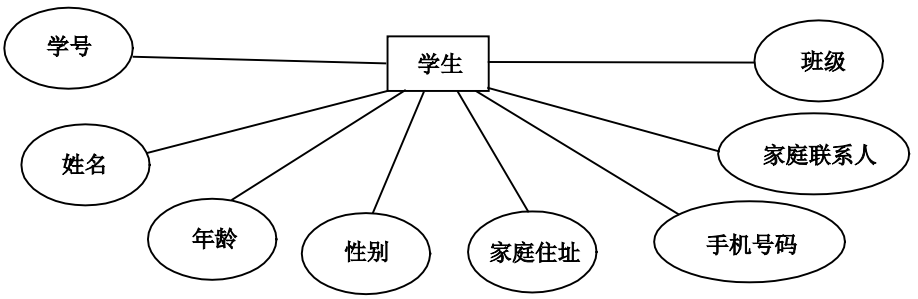


图 1-8　学生实体属性图

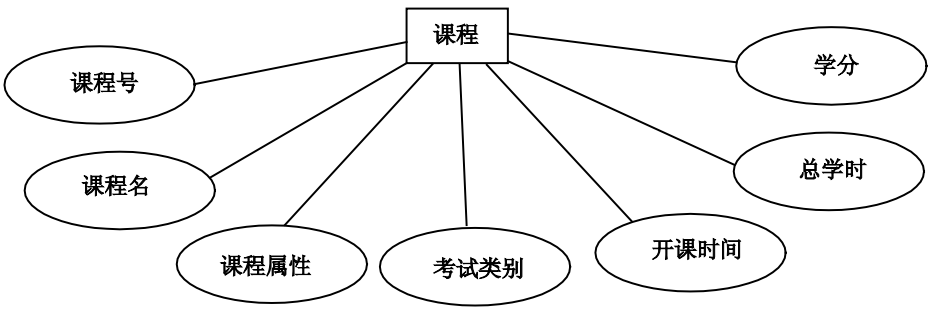


图 1-9　课程实体属性图

再次，定义联系。

院系与辅导员：一个院系有一名辅导员，这名辅导员只能在这个院系中工作，院系与辅导员之间是一对一的联系；

辅导员与班级：一名辅导员管理多个班级，一个班级由一名辅导员管理，辅导员与班级之间是一对多的联系；

班级与学生：一个班级由多名学生组成，由于不能在本校攻读双学位，因此一个学生只能在一个班级就读，班级与学生之间是一对多的联系；

学生与课程：一名学生选修多门课程，一门课程是由多名学生选修，学生与课程之间是多对多的联系。

最后，绘制学生选修系统 E-R 图如图 1-10 所示。

这是一个简单的需求，如果项目较大时，需求量也会随之增加，因此，直接绘制 E-R 图一般比较困难，所以一般先绘制分 E-R 图，再合并分 E-R 图得到总 E-R 图。

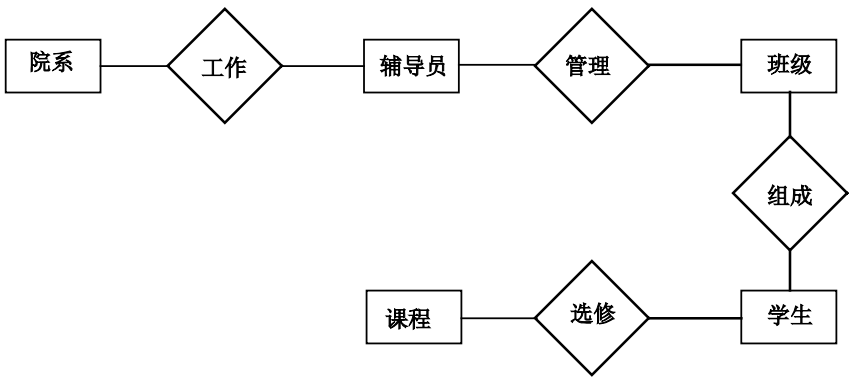


图 1-10　学生选修系统 E-R 图

习题

1. 中集化骏车辆有限公司销售管理系统的需求描述：在全国设立多个销售部，每个销售部里聘用多名职工，每名职工只能在一个销售部工作，在每个销售部可以销售多种车辆类型，每月每个销售部汇总当月销售情况。

试画出 E-R 图，并在图上注明属性、联系的类型。

2. 高校协会管理系统的需求描述：每个高校设立多个协会，每个协会有一名会长，招募多名会员，每个会员可以参加多个协会，每个协会可以举办多个活动，每个活动可以由多个协会共同举办。

试画出 E-R 图，并在图上注明属性、联系的类型。

四、数据库系统结构

考察数据库系统的结构可以有多种不同的层次或不同的角度。

从数据库管理系统角度看，数据库系统通常采用外模式、模式、内模式三级模式结构；这是数据库管理系统内部的系统结构。

从数据库最终用户角度看，数据库系统的结构分为单用户结构、主从式结构、分布式

结构和客户/服务器结构。这是数据库系统外部的体系结构。

1. 数据库系统模式的概念

在数据模型中有“型”（Type）和“值”（Value）的概念。型是指对某一类数据的结构和属性的说明，值是型的一个具体赋值。例如：课程定义为（课程号，课程名，开课学期，考试类别，学分，学时），则是从“型”上记录课程；（C06，数据库系统应用，5，考试，3，68）则是该课程“型”的一个“值”。

模式（Schema）是数据库中全体数据的逻辑结构和特征的描述，它仅仅涉及到型的描述，不涉及到具体的值。模式的一个具体值称为模式的一个实例（Instance）。同一个模式可以有很多实例。模式是相对稳定的，而实例是相对变动的，因为数据库中的数据是在不断更新的。模式反映的是数据的结构及其联系，而实例反映的是数据库某一时刻的状态。虽然实际的数据库管理系统产品种类很多，它们支持不同的数据模型，使用不同的数据库语言，建立在不同的操作系统之上，数据的存储结构也各不相同，但它们在体系结构上通常都具有相同的特征，即采用三级模式结构（早期微机上的小型数据库系统除外）并提供二级映象功能。

2. 数据库系统的三级模式结构

数据库系统的三级模式结构是指数据库系统是由外模式、模式和内模式三级构成。

（1） 模式（Schema）。

模式也称逻辑模式，是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。它是数据库系统模式结构的中间层，既不涉及数据的物理存储细节和硬件环境，也与具体的应用程序，与所使用的应用开发工具及高级程序设计语言（如 C，COBOL，FORTRAN）无关。DBMS 提供模式描述语言（模式 DDL）来严格地定义模式。

一个数据库只有一个模式；模式是数据库数据在逻辑级上的视图；数据库模式以某一种数据模型为基础；定义模式时不仅要定义数据的逻辑结构（如数据记录由哪些数据项构成，数据项的名字、类型、取值范围等），而且要定义与数据有关的安全性、完整性要求，定义这些数据之间的联系。

（2） 外模式（External Schema）。

外模式也称子模式（Subschema）或用户模式，是数据库用户（包括应用程序员和最终用户）能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。DBMS 提供子模式描述语言（子模式 DDL）来严格地定义子模式。

一个数据库可以有多个外模式；外模式就是用户视图；外模式是保证数据安全性的一个有力措施。

（3） 内模式（Internal Schema）。

内模式也称存储模式（Storage Schema），它是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式（例如，记录的存储方式是顺序存储、按照 B 树结构存储还是按 Hash 方法存储；索引按照什么方式组织；数据是否压缩存储，是否加密；数据的存储记录结构有何规定）。DBMS 提供内模式描述语言（内模式 DDL，或者存储模式 DDL）来严格地定义内模式。

一个数据库只有一个内模式；一个表可能由多个文件组成，如：数据文件、索引文件。它是数据库管理系统（DBMS）对数据库中数据进行有效组织和管理的方法，其目的首先是为了减少数据冗余，实现数据共享；其次是为了提高存取效率，改善性能。

数据库系统的三级模式结构如图 1-11 所示。

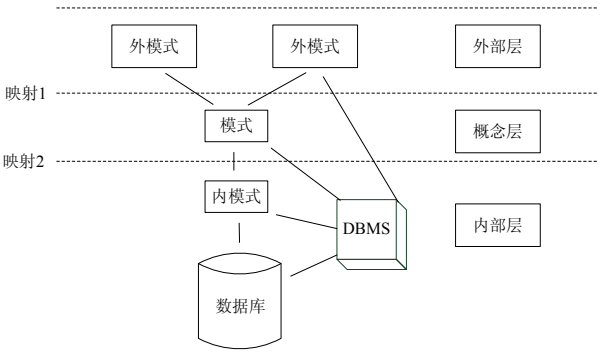


图 1-11　数据库系统的三级模式结构

对数据库系统的综合解读如图 1-12 所示。

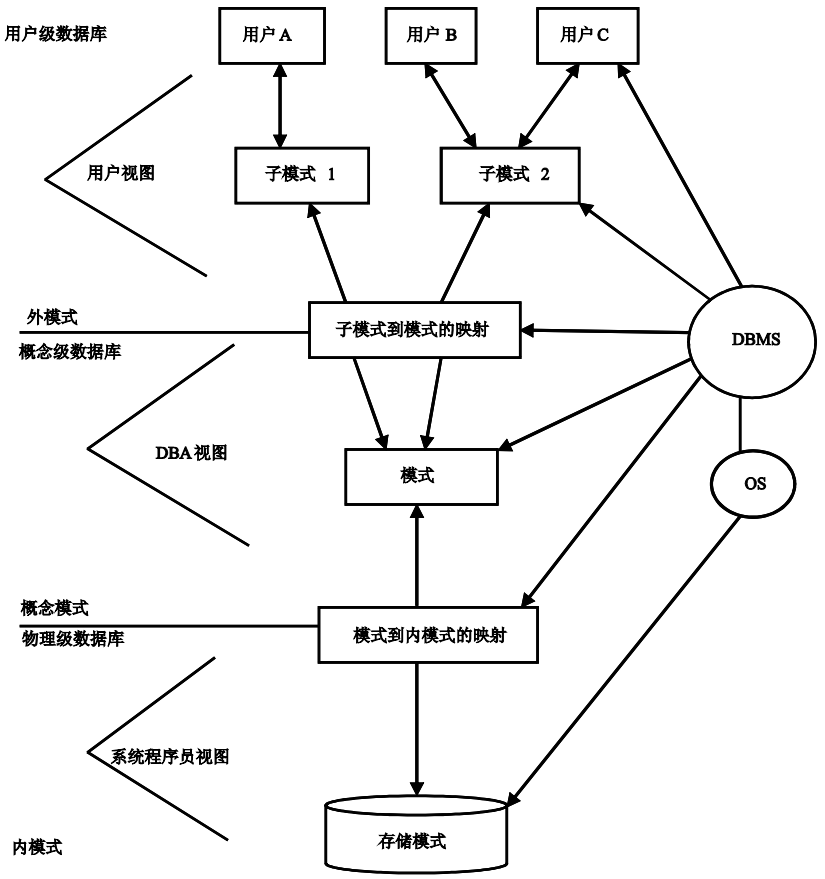


图 1-12　数据库三级模式的综合解读图

3. 数据库的二级映象功能与数据独立性

数据库系统的三级模式是对数据的三个抽象级别，它把数据的具体组织留给 DBMS 管理，使用户能逻辑、抽象地处理数据，而不必关心数据在计算机中的具体表示方式与存储方式。为了能够在内部实现这三个抽象层次的联系和转换，数据库管理系统在这三级模式之间提供了外模式/模式映象和模式/内模式映象两层映象。

正是这两层映象保证了数据库系统中的数据能够具有较高的逻辑独立性和物理独立性。

（1） 外模式/模式映象。

模式描述的是数据的全局逻辑结构，外模式描述的是数据的局部逻辑结构。对应于同一个模式可以有任意多个外模式。对于每一个外模式，数据库系统都有一个外模式/模式映象，它定义了该外模式与模式之间的对应关系。这些映象定义通常包含在各自外模式的描述中。

当模式改变时（例如增加新的关系、新的属性、改变属性的数据类型等），由数据库管理员对各个外模式/模式的映象作相应改变，可以使外模式保持不变。应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性。

（2） 模式/内模式映象。

数据库中只有一个模式，也只有一个内模式，所以模式/内模式映象是唯一的，它定义了数据全局逻辑结构与存储结构之间的对应关系。例如，说明逻辑记录和字段在内部是如何表示的。该映象定义通常包含在模式描述中。当数据库的存储结构改变了（例如选用了另一种存储结构），由数据库管理员对模式/内模式映象作相应改变，可以使模式保持不变，从而应用程序也不必改变。保证了数据与程序的物理独立性，简称数据的物理独立性。

在数据库的三级模式结构中，数据库模式即全局逻辑结构是数据库的中心与关键，它独立于数据库的其他层次。因此设计数据库模式结构时应首先确定数据库的逻辑模式。

数据库的内模式依赖于它的全局逻辑结构，但独立于数据库的用户视图即外模式，也独立于具体的存储设备。它是将全局逻辑结构中所定义的数据结构及其联系按照一定的物理存储策略进行组织，以达到较好的时间与空间效率。

数据库的外模式面向具体的应用程序，它定义在逻辑模式之上，但独立于存储模式和存储设备。当应用需求发生较大变化，相应外模式不能满足其视图要求时，该外模式就得做相应改动，所以设计外模式时应充分考虑到应用的扩充性。

特定的应用程序是在外模式描述的数据结构上编制的，它依赖于特定的外模式，与数据库的模式和存储结构独立。不同的应用程序有时可以共用同一个外模式。数据库的二级映象保证了数据库外模式的稳定性，从而从底层保证了应用程序的稳定性，除非应用需求本身发生变化，否则应用程序一般不需要修改。

数据与程序之间的独立性，使得数据的定义和描述可以从应用程序中分离出去。另外，由于数据的存取由 DBMS 管理，用户不必考虑存取路径等细节，从而简化了应用程序的编制，大大减少了应用程序的维护和修改。

4. 数据库系统的组成

数据库系统一般由数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员和用户构成。下面分别介绍这几个部分的内容。

1） 硬件平台及数据库。

由于数据库系统数据量都很大，加之 DBMS 丰富的功能使得自身的规模也很大，因此整个数据库系统对硬件资源提出了较高的要求，这些要求是：

（1） 要有足够大的内存，存放操作系统、DBMS 的核心模块、数据缓冲区和应用程序。

（2） 有足够大的磁盘等直接存取设备存放数据库，有足够的磁带（或微机软盘）作数据备份。

（3） 要求系统有较高的通道能力，以提高数据传送率。

2） 软件。

数据库系统的软件主要包括：

（1） DBMS。DBMS 是为数据库的建立、使用和维护配置的软件。

（2） 支持 DBMS 运行的操作系统。

（3） 具有与数据库接口的高级语言及其编译系统，便于开发应用程序。

（4） 以 DBMS 为核心的应用开发工具。应用开发工具是系统为应用开发人员和最终用户提供的高效率、多功能的应用生成器、第四代语言等各种软件工具。它们为数据库系统的开发和应用提供了良好的环境。

（5） 为特定应用环境开发的数据库应用系统。

3） 人员。

开发、管理和使用数据库系统的人员主要是：数据库管理员、系统分析员和数据库设计人员、应用程序员和最终用户。不同的人员涉及不同的数据抽象级别，具有不同的数据视图。

任务 2　关系数据库

一、关系数据库描述

关系数据库，是建立在关系数据模型基础上的数据库，借助于集合代数等概念和方法来处理数据库中的数据，同时也是一个被组织成一组拥有正式描述性的表格，该形式的表格作用的实质是装载着数据项的特殊收集体，这些表格中的数据能以许多不同的方式被存取或重新召集而不需要重新组织数据库表格。关系数据库的定义造成元数据的一张表格或造成表格、列、范围和约束的正式描述。每个表格包含用列表示的一个或更多的数据种类。每行包含一个唯一的数据实体，这些数据是被列定义的种类。当创造一个关系数据库的时候，你能定义数据列的可能值的范围和可能应用于那个数据值的进一步约束。而 SQL 语言是标准用户和应用程序到关系数据库的接口。其优势是容易扩充，且在最初的数据库创造之后，一个新的数据种类能被添加而不需要修改所有的现有应用软件。目前主流的关系数据库有 Oracle、SQL Server、DB2、Access、PostgreSQL、Informix 等。

关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。

1. 关系数据结构

关系模型的数据结构非常单一。在关系模型中，现实世界的实体以及实体间的各种联系均用关系来表示。在用户看来，关系模型数据结构是一张二维表，但关系一定是一张二维表，二维表不一定是关系。

（1） 域（Domain）。

定义 1.1 域是一组具有相同数据类型的值的集合。

例如，整数、实数、长度小于 18 个字符的字符串集合、{男，女}、大于等于 0 且小于等于 18 的正整数等，都可以是域。

（2） 笛卡尔积（Cartesian Product）。

定义 1.2 给定一组域 D1，D2，…，Dn，这些域中可以有相同的。D1，D2，…，Dn 的笛卡尔积为：

$$D1\times D2\times\ldots\times Dn=\{(d1，d2，\ldots，dn)\mid di\in Di，i=1，2，\ldots，n\}$$

其中每一个元素（d1，d2，…，dn）叫作一个 n 元组（n-tuple）或简称元组（Tuple）。

元素中的每一个值 di 叫作一个分量（Component）。

笛卡尔积可表示为一个二维表。表中的每行对应一个元组，表中的每列对应一个域。例如给出三个域：

D1=学生集合 STUDENT=刘冰冰，李筱筱

D2=专业集合 SPECIALITY=软件工程专业，古代汉语专业

D3=课程集合 COURSE=数据结构，操作系统，古代汉语，汉语修辞

则 D1，D2，D3 的笛卡尔积为：

D1×D2×D3 ＝｛

(刘冰冰，软件工程专业，数据结构)，(刘冰冰，软件工程专业，操作系统)，

(刘冰冰，软件工程专业，古代汉语)，(刘冰冰，软件工程专业，汉语修辞)，

(刘冰冰，古代汉语专业，数据结构)，(刘冰冰，古代汉语专业，操作系统)，

(刘冰冰，古代汉语专业，古代汉语)，(刘冰冰，古代汉语专业，汉语修辞)，

(李筱筱，软件工程专业，数据结构)，(李筱筱，软件工程专业，操作系统)，

(李筱筱，软件工程专业，古代汉语)，(李筱筱，软件工程专业，汉语修辞)，

(李筱筱，古代汉语专业，数据结构)，(李筱筱，古代汉语专业，操作系统)，

(李筱筱，古代汉语专业，古代汉语)，(李筱筱，古代汉语专业，汉语修辞)｝

其中（刘冰冰，软件工程专业，数据结构）、（李筱筱，古代汉语专业，古代汉语）等都是元组，刘冰冰、软件工程专业、数据结构、李筱筱、古代汉语专业、古代汉语等都是分量。

该笛卡尔积的基数为 2×2×4＝16，也就是说，D1×D2×D3 一共有 2×2×4＝16 个元组。这 16 个元组可列成一张二维表如表 1-1 所示。

表 1-1　学生_专业_课程的笛卡尔积

学　生	专　业	课　程
刘冰冰	软件工程专业	数据结构
刘冰冰	软件工程专业	操作系统
刘冰冰	软件工程专业	古代汉语
刘冰冰	软件工程专业	汉语修辞
刘冰冰	古代汉语专业	数据结构
刘冰冰	古代汉语专业	操作系统
刘冰冰	古代汉语专业	古代汉语
刘冰冰	古代汉语专业	汉语修辞
李筱筱	软件工程专业	数据结构
李筱筱	软件工程专业	操作系统
李筱筱	软件工程专业	古代汉语
李筱筱	软件工程专业	汉语修辞
李筱筱	古代汉语专业	数据结构
李筱筱	古代汉语专业	操作系统
李筱筱	古代汉语专业	古代汉语
李筱筱	古代汉语专业	汉语修辞

（3） 关系（Relation）。

定义 1.3　D1×D2×…×Dn 的子集叫作在域 D1，D2，…，Dn 上的关系，表示为 R（D1，D2，…，Dn）。

这里 R 表示关系的名字，n 是关系的目或度（Degree）。

关系中的每个元素是关系中的元组，通常用 t 表示。

当 n=1 时，称该关系为单元关系（Unary relation）。

当 n=2 时，称该关系为二元关系（Binary relation）。

关系是笛卡尔积的有限子集，所以关系也是一个二维表，表的每行对应一个元组，表的每列对应一个域。由于域可以相同，为了加以区分，必须对每列起一个名字，称为属性（Attribute）。n 目关系必有 n 个属性。

若关系中的某一属性组的值（最小值的集合）能唯一地标识一个元组，则称该属性组为码（key），或称为候选码（Candidate key）。

若一个关系有多个码，则选定其中一个为主码（Primary key）。主码的诸属性称为主属性（Prime attribute）。不包含在任何侯选码中的属性称为非码属性（Non-key attribute）或非主属性。在最简单的情况下，候选码只包含一个属性。在最极端的情况下，关系模式的所有属性组是这个关系模式的候选码，称为全码（All-key）。

一般来说，在域上进行的笛卡尔积运算是没有实际意义的，如 D1×D2×D3（如表 1-1 所示）中，若学生集中没有一个攻读双学位的学生，因此软件工程专业的学生，只能学习本专业的课程数据结构、操作系统，而不可能攻读古代汉语专业的课程古代汉语、汉语修辞；反之亦然。表 1-2 给出学生刘冰冰就读软件工程专业、李筱筱攻读古代汉语专业。

表 1-2　S_P_C 关系

学　生	专　业	课　程
刘冰冰	软件工程专业	数据结构
刘冰冰	软件工程专业	操作系统
李筱筱	古代汉语专业	古代汉语
李筱筱	古代汉语专业	汉语修辞

关系可以有三种类型：基本关系（通常又称为基本表或基表）、查询表和视图表。基本表是实际存在的表，它是实际存储数据的逻辑表示。查询表是查询结果对应的表。视图表是由基本表或其他视图表导出的表，是虚表，不对应实际存储的数据。

关系存在六个性质：列值同域性；列名唯一性；列序无关性；码值唯一性；行序无关性；列值原子性。

2. 关系操作集合

关系模型给出了关系操作的能力，但不对 RDBMS 语言给出具体的语法要求，即不同的 RDBMS 可以定义和开发不同的语言来实现这些操作。

关系模型中常用的关系操作包括：并（Union）、交（Intersection）、差（Difference）、选择（Selecttion）、投影（Project）、连接（Join）、除（Divide）、查询（Query）类操作和增加（Insert）、删除（Delete）、修改（Update）等更新类操作两大部分。查询的表达能力是其中最主要的部分。

关系操作的特点是集合操作方式，即操作的对象和结果都是集合。这种操作方式也称为一次一集合（set-at-a-time）的方式。相应地，非关系数据模型的数据操作方式则为一次一记录（record-at-a-time）的方式。

早期的关系操作能力通常用代数方式或逻辑方式来表示，分别称为关系代数和关系演算。关系代数是用对关系的运算来表达查询要求的方式。关系演算是用谓词来表达查询要求的方式。关系演算又可按谓词变元的基本对象是元组变量还是域变量分为元组关系演算和域关系演算。关系代数、元组关系演算和域关系演算三种语言在表达能力上是完全等价的。关系代数、元组关系演算和域关系演算均是抽象的查询语言，这些抽象的语言与具体的 DBMS 中实现的实际语言并不完全一样。但它们能用作评估实际系统中查询语言能力的标准或基础。实际的查询语言除了提供关系代数或关系演算的功能外，还提供了许多附加功能，例如集函数、关系赋值、算术运算等。

另外还有一种介于关系代数和关系演算之间的语言 SQL（Standard Query Language）即 SQL 语言。SQL 不仅具有丰富的查询功能，而且具有数据定义和数据控制功能，是集查询（Query）、数据定义（DDL）、数据操纵（DML）和数据控制（DCL）于一体的关系数据语言。它充分体现了关系数据语言的特点和优点，是关系数据库的标准语言。

因此，关系数据语言可以分为三类：

（1） 关系代数语言，例如 ISBL（Information System Base Language）。

（2） 关系演算语言分为元组关系演算语言和域关系演算语言，例如 APLHA 为元组关系演算语言的代表，QBE 是域关系演算语言。

（3）具有关系代数和关系演算双重特点的语言，例如结构化查询语言（Structured Query

Language，简称为 SQL）。

这些关系数据语言的共同特点是，语言具有完备的表达能力，是非过程化的集合操作语言，功能强，能够嵌入高级语言中使用。

3. 关系完整性约束

关系模型允许定义三类完整性约束：实体完整性、参照完整性和用户定义的完整性。其中实体完整性和参照完整性是关系模型必须满足的完整性约束条件，应该由关系系统自动支持。用户定义的完整性是应用领域需要遵循的约束条件，体现了具体领域中的语义约束。

1） 实体完整性。

实体完整性指关系中行的完整性，主要用于保证操作的数据非空、唯一且不重复，即实体完整性要求每个关系有且仅有一个主码，每一个主码值必须唯一，且不允许为“空”（NULL）。例如，公民关系中属性“身份证号”为主码；同一学校内学生关系中学号是主码。如有学生信息关系，学号为学生关系的主码，按照实体完整性规则定义如表 1-3 至 1-5 所示，表 1-3 为主码取值正确的关系，表 1-4 和表 1-5 为主码取值错误的情况。

表 1-3　主码取值正确的表

学　号	姓　名	性　别
s1	刘冰冰	女
s2	李筱筱	男
	张灵灵	女
	刘涛	男

表 1-4　主码取空值的表

学　号	姓　名	性　别
s1	刘冰冰	女
s2	李筱筱	男
	张灵灵	女
	刘涛	男

表 1-5　主码取重复值的表

学　号	姓　名	性　别
s1	刘冰冰	女
s2	李筱筱	男
s1	张灵灵	女
s2	刘涛	男

2） 参照完整性。

参照完整性也称为引用完整性。现实世界中的实体之间通常存在着某种联系，在关系模型中，实体以及实体之间的联系都是用关系来表示的，自然而然在关系中存在着实体与实体之间的引用，参照完整性就是描述实体之间这种引用的。

设有学生关系、专业关系分别如表 1-6 和表 1-7 所示，在学生关系中，学号为主码；在专业关系中，专业编号为主码。

表 1-6　学生关系表

学　号	姓　名	性　别	专业编号
s1	刘冰冰	女	p1
s2	李筱筱	男	p1
s3	张灵灵	女	p2
s4	刘涛	男	p3

表 1-7　专业关系表

专业编号	专业名称	专业负责人
p1	软件工程	孙宁宁
p2	计算机科学与技术	汪静静
p3	网络工程	王晓晓

将学生关系与专业关系分别记作：

学生（**学号**，姓名，性别，专业编号）

专业（**专业编号**，专业名称，专业负责人）

带有下划线的属性代表是该关系的主码。

在这两个关系之间存在着属性的引用：学生关系中的属性“专业编号”引用了专业关系中的主码“专业编号”，通过这种引用限制学生关系中的属性“专业编号”的取值必须是专业关系中存在的值，即学生关系中“专业编号”的取值参照了专业关系中的“专业编号”的取值。这种通过一个关系中某属性的取值限制另一个关系中某属性取值的规则称为参照完整性约束。

分析学生关系与专业关系发现，其实在个别高校的专业选择上，大学一年级是不分专业的，通过一年的学习与了解，大二再选择专业，因此学生在大学一年级时期，是没有专业的，这种情况下，学生关系中属性“专业编号”是没有值的，即取空值。

因此，学生关系中属性“专业编号”的取值有两种情况：

（1） 或者值为空；

（2） 或者为专业关系中某个元组的“专业编号”值。

定义 1.4 设 F 是关系 R 的一个或一组属性，但不是关系 R 的码，如果 F 与关系 S 的主码相对应，则称 F 是关系 R 的外码（Foreign Key），并且关系 R 称为参照关系（Referencing Relation），关系 S 为被参照关系（Referenced Relation）或目标关系（Target Relation）。关系 R 和 S 不一定是不同的关系。例如在学生关系与专业关系之间，学生关系称为参照关系，该关系中属性“专业编号”的取值参照了专业关系中主码“专业编号”的取值，专业关系称为被参照关系，将学生关系中“专业编号”称为外码，表示它的取值受限于专业关系中主码的取值。学生关系与专业关系的参照图如图 1-13 所示。

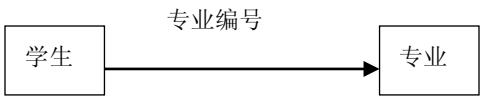


图 1-13 学生专业关系参照图

注意：外码并不一定要与相对应关系的主码同名。但在实际开发中，为了便于识别，当外码与相应的主码属于不同的关系时，一般取相同的名字。

参照完整性规则就是定义外码与被参照的主码之间的引用规则。

对于外码取值，一般应符合如下要求：

（1） 取空值 NULL；

（2） 等于被参照关系中某个元组的主码值。

例如有仓库、产品、库存关系分别如下：

仓库（仓库编号，仓库名，面积，位置，联系电话）

产品（产品编号，产品名称，产品体积，最多堆垛层数，生产日期，保质期）

库存（产品编号，仓库编号，入库日期，库存量）

分析这三个关系：关系库存中属性“产品编号”引用了关系产品中的码“产品编号”，同时关系库存中属性“仓库编号”引用了关系仓库中码“仓库编号”，因此，关系库存中属性“产品编号”和“仓库编号”是外码，关系库存是参照关系，关系仓库与关系产品均是被参照关系。根据参照完整性规则对外码取值的约定，关系库存中属性“产品编号”和“仓库编号”的取值有两种：空值或被参照关系中某个组的主码值。

进一步分析关系库存发现，该关系的主码为组合属性（产品编号，仓库编号，入库日期），按照实体完整性规则，该关系的主码取值不能取空值或者重复的值。

综合以上得出：关系库存中外码“产品编号”和“仓库编号”的取值仅仅限于被参照关系中某个组的主码值。

3） 用户定义的完整性。

用户定义的完整性也称为域完整性或语义完整性，它是指明关系中属性的取值范围，也就是属性的域，即限制关系中的属性的取值类型及取值范围，防止属性的值与应用语义矛盾。用户定义的完整性是针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求。如在高校学生信息管理系统中，性别取{男，女}、学生年龄（单位：岁）取[10，50]、姓名最长取 50 个字符、身高（单位：米）取[0.45，2.80]等。

二、关系代数

关系代数是一种抽象的查询语言，用对关系的运算来表达查询，作为研究关系数据语言的数学工具。

关系代数的运算对象是关系，运算结果也为关系。关系代数用到的运算符包括传统的集合运算符和专门的关系运算符两类。因此，按照运算符的不同，将关系代数的运算分为传统的集合运算和专门的关系运算两类，传统的集合运算有并、差、交和笛卡尔积 4 个，它们都是双目运算符；专门的关系运算有选择、投影、连接和除 4 个，选择和投影是单目运算符，连接和除是双目运算符。关系运算中的运算符如表 1-8 所示。

表 1-8　关系运算符

运 算 符		含　义
传统的集合运算符	∪	并
	∩	交
	−	差
	×	笛卡尔积
专门的关系运算符	σ	选择
	π	投影
	∞	连接
	÷	除

关系代数运算中，所用到的抽象关系如图 1-14 所示。

A	B
a1	b1
a2	b2
a3	b2

（a）关系 R

A	B
a1	b1
a1	b2
a2	b1
a2	b2

（b）关系 S

A	C	D
a1	c1	d1
a1	c2	d1
a2	c2	d2
a1	c1	d2

（c）关系 T

图 1-14　关系代数运算中用到的抽象关系图

关系代数运算中，所用到的实际例子如图 1-15 所示。

sid	sName	sAge
1334120101	刘冰冰	17
1334120102	李筱筱	18
1334110101	李思涵	20
1334110102	刘明明	19

（a）所有专业学生关系 student1

sid	sName	sAge
1334120101	刘冰冰	17
1334120102	李筱筱	18
1434120101	王帅中	19
1434120102	李阳阳	18

（b）软工专业学生关系 student2

cid	cName	cCredit
1	算法分析与设计	2
2	操作系统	3
3	数据库系统原理	3

（c）课程关系 course

sid	cid	totalScore
1334120101	1	89
1334120102	1	98
1334110101	1	90
1334110102	1	95
1334120101	2	88
1334120102	2	90
1334110102	2	86
1334120101	3	85
1334120102	3	91
1334110101	3	96
1434120101	1	79
1434120102	1	95
1434120102	2	80
1434120101	3	70

（d）成绩关系 grade

图 1-15　关系运算中所用到的实际关系

1. 传统的集合运算

（1） 并（Union）。

设关系 R 和关系 S 具有相同的目 n（即两个关系都有 n 个属性），且相应的属性取自同一个域，则关系 R 与关系 S 的并由属于 R 或属于 S 的元组组成。其结果关系仍为 n 目关系。记作：

$$R \cup S=\{t|t \in R \vee t \in S\}$$

对图 1-14 中的关系 R 与关系 S 执行并运算后，所得结果如图 1-16 所示。

关系 R∪S

A	B
a1	b1
a2	b2
a3	b2
a1	b2
a2	b1

图 1-16　并运算结果

例如，对图 1-15 所示的 13 级学生信息表与软工专业学生信息表执行并运算 student1 ∪student2 后，得到的结果是 13 级或者软工专业学生信息综合表，如图 1-17 所示。

sid	sName	sAge
1334120101	刘冰冰	17
1334120102	李筱筱	18
1334110101	李思涵	20
1334110102	刘明明	19
1434120101	王帅中	19
1434120102	李阳阳	18

图 1-17　13 级或者软工专业学生信息综合表

（2） 差（Difference）。

设关系 R 和关系 S 具有相同的目 n，且相应的属性取自同一个域，则关系 R 与关系 S 的差由属于 R 而不属于 S 的所有元组组成。其结果关系仍为 n 目关系。记作：

$$R-S=\{t|t\in R\wedge t\notin S\}$$

对图 1-14 中的关系 R 与关系 S 执行差运算后，所得结果如图 1-18 所示。

关系 R−S

A	B
a3	b2

图 1-18　差运算结果

例如，图 1-15 中的 13 级学生信息表对软工专业学生信息表执行差运算 student1−student2 后，得到的结果为 13 级非软工专业学生信息表，如图 1-19 所示。

sid	sName	sAge
1334110101	李思涵	20
1334110102	刘明明	19

图 1-19　13 级非软工专业学生信息表

（3） 交（Intersection Referential Integrity）。

设关系 R 和关系 S 具有相同的目 n，且相应的属性取自同一个域，则关系 R 与关系 S 的交由既属于 R 又属于 S 的元组组成。其结果关系仍为 n 目关系。记作：

$$R\cap S=\{t|t\in R\wedge t\in S\}$$

对图 1-14 中的关系 R 与关系 S 执行交运算后，所得结果如图 1-20 所示。

关系 R∩S

A	B
a1	b1
a2	b2

图 1-20　交运算结果

例如，对图 1-15 中 13 级学生信息表与软工专业学生信息表执行交运算 student1∩student2 后，得到的结果是 13 级软工专业学生信息表，如图 1-21 所示。

sid	sName	sAge
1334120101	刘冰冰	17
1334120102	李筱筱	18

图 1-21　13 级软工专业学生信息表

（4） 笛卡尔积（Cartesian Product）。

两个分别为 n 目和 m 目的关系 R 和 S 的广义笛卡尔积是一个(n+m)列的元组的集合。元组的前 n 列是关系 R 的一个元组，后 m 列是关系 S 的一个元组。若 R 有 k1 个元组，S 有 k2 个元组，则关系 R 和关系 S 的笛卡尔积有 k1×k2 个元组。

对图 1-14 中的关系 R 与关系 T 执行笛卡尔积运算后，所得结果如图 1-22 所示。

关系 R×T

R.A	R.B	T.A	T.C	T.D
a1	b1	a1	c1	d1
a1	b1	a1	c2	d1
a1	b1	a2	c2	d2
a1	b1	a1	c1	d2
a2	b2	a1	c1	d1
a2	b2	a1	c2	d1
a2	b2	a2	c2	d2
a2	b2	a1	c1	d2
a3	b2	a1	c1	d1
a3	b2	a1	c2	d1
a3	b2	a2	c2	d2
a3	b2	a1	c1	d2

图 1-22　笛卡尔积运行结果

例如，对 13 级学生信息表与课程表执行广义笛卡尔积 student1×course 后，得到的结果如图 1-23 所示。

sid	sName	sAge	cid	cName	cCredit
1334120101	刘冰冰	17	1	算法分析与设计	2
1334120102	李筱筱	18	1	算法分析与设计	2
1334110101	李思涵	20	1	算法分析与设计	2
1334110102	刘明明	19	1	算法分析与设计	2
1334120101	刘冰冰	17	2	操作系统	3
1334120102	李筱筱	18	2	操作系统	3
1334110101	李思涵	20	2	操作系统	3
1334110102	刘明明	19	2	操作系统	3
1334120101	刘冰冰	17	3	数据库系统原理	3
1334120102	李筱筱	18	3	数据库系统原理	3
1334110101	李思涵	20	3	数据库系统原理	3
1334110102	刘明明	19	3	数据库系统原理	3

图 1-23　13 级学生信息表与课程表执行笛卡尔运算的结果

从图 1-23 的运算结果看出，笛卡尔运算的结果非常庞大，并且通常有些数据是没有任

何意义的，因此，一般该运算尽可能少用或者不用。

2. 专门的关系运算

为了叙述上的方便，我们先引入几个记号。

设关系模式为 R（$A_1, A_2, \ldots, A_n$）。它的一个关系设为 R。t∈R 表示 t 是 R 的一个元组。t[A_i]则表示元组 t 中相应于属性 A_i 的一个分量。

若 A={$A_{i1}, A_{i2}, \ldots, A_{ik}$}，其中 $A_{i1}, A_{i2}, \ldots, A_{ik}$ 是 $A_1, A_2, \ldots, A_n$ 中的一部分，则 A 称为属性列或域列。$\overline{A}$ 则表示{$A_1, A_2, \ldots, A_n$}中去掉{$A_{i1}, A_{i2}, \ldots, A_{ik}$}后剩余的属性组。t[A]=(t[$A_{i1}$], t[$A_{i2}$], …, t[$A_{ik}$])表示元组 t 在属性列 A 上诸分量的集合。

R 为 n 目关系，S 为 m 目关系。设 $\mathbf{t_r}$∈R，$\mathbf{t_s}$∈S，则 $\overset{\frown}{t_r\ t_s}$ 称为元组的连接（Concatenation）。它是一个（n+m）列的元组，前 n 个分量为 R 中的一个 n 元组，后 m 个分量为 S 中的一个 m 元组。

定义 1.5　给定一个关系 R(X,Z)，X 和 Z 为属性组。我们定义，当 t[X]=x 时，x 在 R 中的象集（Images Set）为：

$$Z_x=\{t[Z]|t\in R, t[X]=x\}$$

它表示 R 中属性组 X 上值为 x 的诸元组在 Z 上分量的集合。

例如在图 1-14 的关系 R 中，a_1 在关系 R 中的象集 Z_{a1}={(c1，d1)，(c2，d1)，(c1，d2)}，a2 在关系 R 中的象集 Z_{a2}={(c2，d2)}。

（1） 选择（Selection）。

选择又称为限制（Restriction）。它是在关系 R 中选择满足给定条件的诸元组，记作：

$$\sigma F(R) = \{t|t\in R \ \wedge \ F(t)='真'\}$$

其中 F 表示选择条件，它是一个逻辑表达式，取逻辑值“真”或“假”。

逻辑表达式 F 的基本形式为：

$$X_1\ \theta\ Y_1\ [\ \varphi\ X_2\ \theta\ Y_2\ ]$$

θ 表示比较运算符，它可以是>、≥、<、≤、=或≠。X_1、Y_1 等是属性名或常量或简单函数。属性名也可以用它的序号来代替。φ 表示逻辑运算符，它可以是!、∧或∨。[ ]表示可选项，即[ ]中的部分可以要也可以不要，表示上述格式可以重复下去。

因此选择运算实际上是从关系 R 中选取使逻辑表达式 F 为真的元组。这是从行的角度进行的运算。

对图 1-14 中关系 R 取出 C 等于 c1 的运算如下：

$$\sigma_{C=c1}(T)$$

运算结果如图 1-24 所示。

对关系 T 执行选出 C=c1 后的关系

A	C	D
a1	c1	d1
a1	c1	d2

图 1-24　选择运算结果

例如，对图 1-15 中 13 级学生信息表执行运算如下：

$$\sigma_{sAge=17 \vee sAge=18}(student1)$$

其功能是从 13 级学生信息表中查询年龄为 17 岁或者年龄为 18 岁的学生信息，运算结果如图 1-25 所示。

sid	sName	sAge
1334120101	刘冰冰	17
1334120102	李筱筱	18

图 1-25　选择运算结果

（2） 投影（Projection）。

关系 R 上的投影是从 R 中选择出若干属性列组成新的关系，记作：

$$\Pi_A(R) = \{ t[A] \mid t \in R \}$$

其中，A 为 R 中的属性列。

对关系图 1-14 中的关系 R 执行选出 A 列和 C 列的运算如下：

$$\pi_{A,C}(T)$$

对关系 T 执行选出 A 列和 C 列后的新关系，如图 1-26 所示。

A	C
a1	c1
a1	c2
a2	c2

图 1-26　投影运算结果

投影运算是从列的角度对关系进行运算，但实际运算中，对某个关系进行执行投影运算后，不仅取消了原关系中的某些列，同时按照关系行中数据不重复的原则，再取消新关系中完全相同的行，如在本例中，因为在运算结果中存在两行（a1,c1），因此，取消一行，新关系中数据为三行。

例如，对图 1-15 中 13 级学生信息表 student1 执行投影运算如下：

$$\pi_{sid,sName}(student1)$$

其功能是从 13 级学生信息表中查询学生学号及学生姓名信息，运算结果如图 1-27 所示。

sid	sName
1334120101	刘冰冰
1334120102	李筱筱
1334110101	李思涵
1334110102	刘明明

图 1-27　投影运算结果

（3） 连接（Join）。

连接包括 θ 连接、自然连接、外连接、半连接，它是从两个关系的笛卡尔积中选取属性间满足一定条件的元组。

连接运算从 R 和 S 的笛卡尔积 R×S 中选取 R 关系中在 A 属性组上的值与 S 关系中在

B 属性组上值满足比较关系 θ 的元组，记作：

$$R \underset{A\theta B}{\bowtie} S = \{\widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A]\theta t_s[B]\}$$

连接运算中有两种最为重要也最为常用的连接，一种是等值连接（Equi-join），另一种是自然连接（Natural join）。

θ 为“=”的连接运算称为等值连接，它是从关系 R 与 S 的笛卡尔积中选取 A、B 属性值相等的那些元组，记作：

$$R \underset{A=B}{\bowtie} S = \{\widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B]\}$$

自然连接（Natural join）是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组，并且要在结果中把重复的属性去掉，即如果关系 R 与关系 S 中具有相同的属性组 E，A 为两个关系的全体属性集合，则自然连接可记作：

$$R \bowtie S = \{\widehat{t_r t_s}\,[E - A] \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B]\}$$

一般的连接操作是从行的角度进行的。但自然连接还需要取消重复列，所以是同时从行和列的角度进行的。

例如，对图 1-14 中的关系 R 和关系 T，执行非等值连接

$$R \underset{R.A<>T.A}{\bowtie} T$$

的结果如图 1-28 所示。

R.A	B	T.A	C	D
a1	b1	a2	c2	d2
a2	b2	a1	c1	d1
a2	b2	a1	c2	d1
a2	b2	a1	c1	d2
a3	b2	a1	c1	d1
a3	b2	a1	c2	d1
a3	b2	a2	c2	d2
a3	b2	a1	c1	d2

图 1-28　非等值连接运算结果

对图 1-14 中的关系 R 和关系 T，执行等值连接

$$R \underset{R.A=T.A}{\bowtie} T$$

的结果如图 1-29 所示。

R.S	B	T.A	C	D
a1	b1	a1	c1	d1
a1	b1	a1	c2	d1
a1	b1	a1	c1	d2
a2	b2	a2	c2	d2

图 1-29　等值连接运算结果

对图 1-14 中的关系 R 和关系 T，执行自然连接R ⋈ T的结果，如图 1-30 所示。

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d1
a1	b1	c1	d2
a2	b2	c2	d2

图 1-30　自然连接结果

从本例看出，关系 R 与关系 T 做自然连接时，选择两个关系在公共属性上值相等的元组构成新的关系，但关系 R 中的元组（a3,b2）在关系 T 中不存在公共属性相等的元组，从而造成了该元组在执行运算的过程中被舍弃的现象，这些被舍弃的元组称为悬浮元组。如果把悬浮元组也保存在结果关系中，而在其他属性上填空值（NULL），那么这种连接叫做外连接。如对关系 R 和关系 T 执行外连接运算的结果，如图 1-31 所示。

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d1
a1	b1	c1	d2
a2	b2	c2	d2
a3	b2	NULL	NULL

图 1-31　外连接运算结果

例如，对图 1-15 中 13 级学生信息表 student1 与成绩表 grade 执行如下操作：

$$R \underset{\text{student1.sid=grade.sid}}{\bowtie} S$$

其功能是查询 13 级学生的学号、姓名、年龄、选修课程编号及成绩，结果如图 1-32 所示。

sid	sName	sAge	cid	totalscore
1334120101	刘冰冰	17	1	89
1334120102	李筱筱	18	1	98
1334110101	李思涵	20	1	90
1334110102	刘明明	19	1	95
1334120101	刘冰冰	17	2	88
1334120102	李筱筱	18	2	90
1334110102	刘明明	19	2	86
1334120101	刘冰冰	17	3	85
1334120102	李筱筱	18	3	91
1334110101	李思涵	20	3	96

图 1-32　选择投影运算结果

（4） 除（Division）。

给定关系 R(X，Y)和 S(Y，Z)，其中 X, Y, Z 为属性组。R 中的 Y 与 S 中的 Y 可以有不同的属性名，但必须出自相同的域集。R 与 S 的除运算得到一个新的关系 P(X)。该 P 中只包含 R 中投影下来的的 X 属性组，且该 X 属性组应满足：R(Y)=S(Y)。

例如，设关系 R 与关系 S 如图 1-33 所示。

关系 R

A	B	C
a1	b1	c1
a1	b2	c1
a2	b2	c2

关系 S

B	D
b1	d1
b1	d2
b1	d3
b2	d1
b2	d3

图 1-33　除运算关系

按定义对应到本例，关系 R(A，B，C)和 S(B，D)中 R 与 S 是根据 B 属性关联起来的，X 为关系模式 R 中除去属性 B 的属性集合——A 属性和 C 属性的组合，而对于关系 S 中的属性 D 是没有任何关系的，可以不予考虑。

属性集(a1，c1)在关系 R 中的象集 $Z_{(a1,\ c1)}$={b1，b2}。

属性集(a2，c2)在关系 R 中的象集 $Z_{(a1,\ c2)}$={b2}。

关系 S 在 B 上的投影为选择 S 上的 B 属性列，其结果为(b1，b2)。

因此，属性集(a1，c1)在关系 R 上的象集满足对(b1，b2)的包含关系，即 R÷S={a1，c1}。

例如，对学生成绩表 grade 除课程表 course 运算：

$$\pi_{\text{sid,cid}}(\text{grade}) \div \text{course}$$

其功能为查询 13 级学生中选修所有课程的学生学号，运算结果如图 1-34 所示。

sid
1334120101
1334120102

图 1-34　除运算结果

3. 关系的查询处理

查询处理是关系数据库管理系统执行查询语句的过程，其任务是把用户提交给关系数据库管理系统的查询语句转换为高效的查询执行计划。查询处理可分为 4 个阶段：查询分析、查询检查、查询优化和查询执行。

查询分析是对查询语句进行研究扫描、从词法和语法进行分析，判断查询语句是否符合相关语法规则，如果没有错误，则进行下一步处理，否则报告查询语句中出现的语法错误并结束。

查询检查是对合法的查询语句进行语义检查，即相应的操作对象是否存在、是否违背完整性约束等。如果正确，则进行下一步处理，否则，报告查询语句中出现的语义错误并结束。

查询优化是选择一个高效执行的查询处理策略，按照优化的层次一般可将查询优化分为代数优化和物理优化。代数优化是指关系代数表达式的优化，按照一定的规则，通过对关系代数表达式进行等价变换，改变代数表达式中操作的次序和组合，使查询执行更高效；物理优化则是指存取路径和底层操作算法的选择。

查询执行是依据优化器得到的执行策略生成查询执行计划，由代码生成器生成执行这个查询计划的代码，然后加以执行，回送查询结果。

如某高校有教务管理系统后台数据库 ssms，其中有学生表、课程表、成绩表，学生表中有学生总人数 20000 人，课程表中有全校开出课程共计 2000 门，每个在校生已经选修的课程有 200000 条，因此，如果有以下三个查询语句：

语句 1 $\pi_{sname,cname,grade}(\delta_{student.sid=score.sid \wedge course.cid=score.cid \wedge grade\geqslant 90}(student\times score\times course))$

语句 2 $\pi_{sname,cname,grade}(\delta_{grade\geqslant 90}(student \bowtie score \bowtie course))$

语句 3 $\pi_{sname,cname,grade}(\delta_{cid}(student \bowtie \delta_{grade\geqslant 90}(score) \bowtie course))$

这三个语句的功能同样都是查询成绩是 90 分以上（包括 90 分）的学生姓名、课程名以及成绩，但其运算量有较大区别：

语句 1 的运算量是：

先将三个表进行笛卡尔运算生成中间表，其数据量是：

$$20000\times 2000\times 200000=8000000000000$$

从八万亿条数据中再查询出符合上述条件的数据。

语句 2 的运算量是：

先将三个表进行自然连接，由于表 score 中是学生选修课程的成绩，因此，自然连接的结果与表 score 中数据量相同，也是 200000，接下来从这 20 万条数据中再检索所需要的数据。

语句 3 的运算量是：

先从表 score 中检索成绩是 90 分以上（包括 90 分）的学生成绩信息，假设成绩中符合条件的是 5%，则其自然连接后结果有 10000 条，后续工作是从这 1 万条数据取出相应列即可。

从以上的分析得知，如果在关系代数中需要做连接操作，则能使用自然连接不使用笛卡尔积运算；能在自然连接前使用选择减少数据则先执行选择，后连接。

习题

1. 关系 R 和 S 如下：

R

A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	12

S

B	E
b1	3
b2	7
b3	10
b4	2
b5	2

求：（1）$R \bowtie_{C<E} S$。

（2）$R \bowtie S$。

2. 设有如图所示的关系 S、C 和 SC，试用关系代数表达式表示下列查询语句：

S

S#	SNAME	AGE	SEX
1	李强	23	男
2	刘丽	22	女
5	张友	22	男

C

C#	CNAME	TEACHER
k1	C 语言	杨英
k5	数据库原理	张玲
k8	编译原理	张玲

SC

S#	C#	GRADE
1	k1	83
2	k1	85
5	k1	92
2	k5	90
5	k5	84
5	k8	80

（1） 检索“张玲”老师所授课的课程号（C#）和课程名（CNAME）。
（2） 检索年龄大于 21 的男学生学号（S#）和姓名（SNAME）。
（3） 检索“李强”同学不学课程的课程号（C#）。
（4） 检索全部学生都选修的课程的课程号（C#）和课程名（CNAME）。
（5） 检索选修全部课程的学生姓名（SNAME）。
（6） 检索选修课程名为“C 语言”的学生学号（S#）和姓名（SNAME）。
（7） 检索没有一门课程成绩不及格的学生的学号（S#）和姓名（SNAME）。

任务 3　关系数据理论

人们在设计数据库的实践中，根据不同的设计方法中出现操作异常和数据冗余的程度，将建立关系需要满足的约束条件划分成若干标准，这些标准称为范式（或者数据库设计范式）。关系数据库中的关系必须满足一定的要求，即满足不同的范式。目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、Boyce-Codd 范式（BCNF）、第四范式（4NF）和第五范式（5NF），范式越高，发生操作异常的可能性越小，数据冗余越小，但由于关联多，读取数据时花费的时间也会相应增加。

满足最低要求的范式是第一范式（1NF）。在第一范式的基础上进一步满足更多要求称为第二范式（2NF），其余范式以此类推。一般说来，数据库只需满足第三范式（3NF）或者 BCNF 就行了。

下面我们举例介绍第一范式（1NF）、第二范式（2NF）、第三范式（3NF）和 BCNF 如表 1-9 所示。

表 1-9　Student_Coure 表

sno	sname	sex	age	class	学生选修课程成绩信息					
					cno	cname	ccredit	tno	tname	grade
s1	张三	男	18	软工 1301B	c1	数据库	4	t1	李白	95
					c2	网络工程	4	t1	李白	89
					c3	软件工程	5	t3	刘邦	98
s2	李丽	女	18	软工 1301B	c1	数据库	4	t1	李白	95
					c3	软件工程	5	t3	刘邦	94
s3	张三	女	17	计科 1301B	c1	数据库	4	t2	杜甫	71
					c2	网络工程	4	t3	刘邦	78
s4	李四	男	18	计科 1301B	c1	数据库	4	t2	杜甫	93
					c3	软件工程	5	t2	杜甫	77

1. 把表规范为关系

首先，由于关系一定是表，但表不一定是关系。按照关系的判定“每一个属性必须是不可再分的”来看，即从该表的最下边往上看，表中每条表格线是否均能到达表头的边框线来判断，Student_Course 表不是一个关系，因为存在有学生选修课程成绩信息属性是再分属性，它是由课程编号 cno、课程名 cname、学分 ccredit、教师编号 tno、教师姓名 tname、成绩 grade 六个属性组成，故将其规范化为一个关系如表 1-10 所示。

表 1-10　关系 Student_Course

sno	sname	sex	age	class	cno	cname	ccredit	tno	tname	grade
s1	张三	男	18	软工1301B	1	数据库	4	t1	李白	95
					2	网络工程	4	t1	李白	89
					3	软件工程	5	t3	刘邦	98
s2	李丽	女	8	软工1301B	1	数据库	4	t1	李白	95
					3	软件工程	5	t3	刘邦	94
s3	张三	女	17	计科1301B	1	数据库	4	t2	杜甫	71
					2	网络工程	4	t3	刘邦	78
s4	李四	男	8	计科1301B	1	数据库	4	t2	杜甫	93
					3	软件工程	5	t2	杜甫	77

如果关系 Student_Course 的所有属性都是不可分的基本数据项，则该关系属于 1NF，记作：Student_Course∈1NF。

2. 查看关系中存在的问题

（1） 数据冗余。

比如，在该关系中，每门课程的相关信息会随着选修该课程的人数变化，如果该课程有 500 人选修，则该课程信息会重复 500 次；另外，如果一个学生选修了 8 门课程，则其个人信息也会重复出现 8 次，这种现象对存储空间来说是一个巨大的浪费。

（2） 插入异常。

若一个学生是刚报到的新生，该生没有了解过任何课程，也没有选修任何课程，则该学生的信息不能存到该数据库。

（3） 修改异常。

如果软件工程这门课程的学分需要修改为 3 学分，如果有 220 个学生选修该课程，则必须修改这 220 个学生所对应的该课程的学分信息，最终导致数据的维护代价较大。

（4） 删除异常。

如果 13 级选修数据库课程的学生在 2017 年 6 月份毕业了，则在删除这批学生的同时，所有与其相关的课程信息也“被”删除了。

从以上的分析可以发现，由于关系 Student_Course 存在大量的数据冗余、插入异常、修改异常、删除异常等问题，因此它不是一个“好”的关系。一个“好”的关系，应该降低数据冗余、并且不存在插入异常、修改异常、删除异常等问题。

3. 产生这些问题的原因

从所给出的数据可以看出，只要给出学生的学号 sno，即可以确定一个学生的（sname，sex，age，class）等信息，因此，将这种由学号 sno 确定关系中其他属性（sname，sex，age，class）称为函数依赖，记作：

$$\text{sno} \rightarrow \text{sname}，\ \text{sno} \rightarrow \text{sex}，\ \text{sno} \rightarrow \text{age}，\ \text{sno} \rightarrow \text{class}$$

合并记作：

$$\text{sno} \rightarrow (\text{sname}，\ \text{sex}，\ \text{age}，\ \text{class})$$

同样，对于该关系中的课程信息，也存在类似的函数依赖如下：

$$\text{cno} \rightarrow (\text{cname}，\ \text{ccredit})$$

授课教师信息中，也存在类似的函数依赖如下：

$$\text{tno} \rightarrow \text{tname}$$

定义 1.6　设 R(U)是属性集 U 上的关系模式。X，Y 是 U 的子集，若对于 R(U)的任意一个可能的关系 r，r 中不可能存在两个元组在 X 上的属性值相等，而在 Y 上的属性值不等，则称 X 函数确定 Y 或 Y 函数依赖于 X，记作：$X \rightarrow Y$。

对于该关系中的成绩属性 grade 来说，由于一个学生选修多门课程，因此给定学号 sno 是不能唯一确定一项成绩的；同样，由于一门课程是由多个学生选修的，因此也不能由课程号 cno 来唯一地确定一项成绩，由于一位教师教授多门课程，因此，给定一个教师编号也不能唯一地确定一项成绩，因此，该属性必须由多个属性共同决定。由于一个学生可以选修多门课程，一门课程可以由多个学生选修，学生与课程之间形成多对多的关系，在建立这种关系的同时，新增一个属性成绩 grade，因此，成绩 grade 的函数依赖是：

$$(\text{sno}，\ \text{cno}) \rightarrow \text{grade}$$

由上边所列的函数依赖确定关系中的码为(sno，cno，tno)。分析该码：对于每一个教师编号，如果使用相应的(sno，cno)是否可以唯一确定呢？从每个学生选修某门课程的关系可以得出，一旦某位学生选修了某门课程，该课程的授课教师已经确定，因此，可以得出这样的函数依赖：

$$(\text{sno}，\ \text{cno}) \rightarrow \text{tno}，\ \text{tno} \rightarrow \text{teacher}$$

所以，关系 Student_Coure 的候选码为(sno，cno)，该关系中存在的函数依赖是：

$$(\text{sno}，\ \text{cno}) \rightarrow (\text{sname}，\ \text{sex}，\ \text{age}，\ \text{class}，\ \text{cname}，\ \text{ccredit}，\ \text{teacher}，\ \text{grade})$$

由主属性的定义“能从函数确定所有属性的最小属性组”得出该关系中的主属性为(sno，cno)，非主属性为(sname，sex，age，class，cname，ccredit，teacher，grade)。

在该关系的函数依赖中，存在如下几类函数依赖：

（1）　完全函数依赖。

对于成绩 grade 属性，它是由(sno，cno) → grade函数确定的，由于在该函数依赖中，不管是学号 sno，还是课程号 cno，都是必不可少的，这种函数依赖称为完全函数依赖（Full Functional Dependency），记作：

$$(\text{sno}，\ \text{cno}) \xrightarrow{F} \text{grade}$$

定义 1.7　在 R(U)中，如果$X \rightarrow Y$，并且对于 X 的任何一个真子集X'，都有$X' \not\rightarrow Y$，则

称 Y 对 X 完全函数依赖，记作：$X \xrightarrow{F} Y$。

（2） 部分函数依赖。

对于学生的姓名 sname、性别 sex、年龄 age、班级 class 属性，既然关系的码是(sno，cno)，因此，其函数依赖是：

$$(\text{sno，cno}) \rightarrow (\text{sname，sex，age，class})$$

由上述分析过程发现，实际上，这些属性由学号 sno 单独确定，有没有课程编号对于学生个人信息的确定没有任何影响，因此，这种函数依赖称为部分函数依赖（Partial Functional Dependency），记作：

$$(\text{sno，cno}) \xrightarrow{P} (\text{sname，sex，age，class})$$

定义 1.8 在 R(U)中，如果$X \rightarrow Y$，但 Y 不完全函数依赖于 X，则称 Y 对 X 部分函数依赖，记作，记作：$X \xrightarrow{P} Y$。

（3） 传递函数依赖。

对于教师编号和教师姓名而言，存在$(\text{sno，cno}) \rightarrow \text{tno}$，$\text{tno} \rightarrow \text{teacher}$函数依赖，这种类型的函数依赖称为传递依赖（Transitive Functional Dependency），记作：

$$(\text{sno，cno}) \xrightarrow{T} \text{teacher}$$

关系 Student_Course 的函数依赖图如图 1-35 所示。

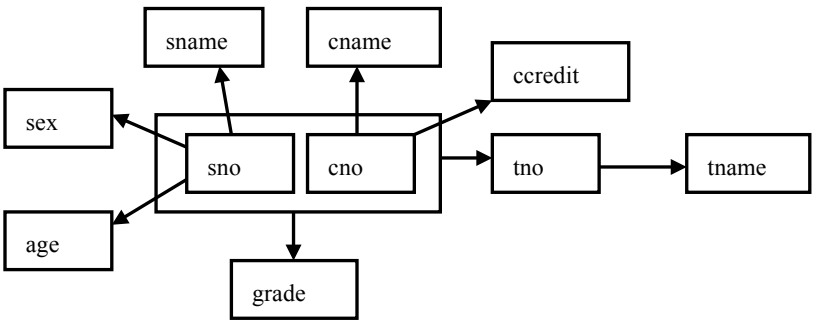


图 1-35 关系 Student_Course 的函数依赖图

定义 1.9 在 R(U)中，如果$X \rightarrow Y$且$Y \not\subseteq X$，$Y \not\rightarrow X$，$Y \rightarrow Z$，$Z \not\subseteq Y$，则称 Z 对 X 传递函数依赖，记作：$X \xrightarrow{T} Y$。

从对函数依赖的分类看出，若在关系中存在的是完全函数依赖，则对比在关系中存在部分函数依赖或者传递函数依赖来说是一个“好”的关系。

4. 分解关系

在关系分解的过程中，一般遵循的原则是先解决部分函数依赖，再解决传递函数依赖。

对于关系 Student_Course，由于学生的姓名 sname、性别 sex、年龄 age、班级 class 属

性和课程的课程名 cname、学分 ccredit 属性均部分函数依赖于码，因此，分解该关系得到关系学生 Student、关系课程 Course、关系选修 S_C，三个关系分别表示为（加下划线的属性表示是该关系的候选码）：

Student(sno，sname，sex，age，class)

Course(cno，cname，ccredit)

S-C(sno，cno，grade，tno，tname)

通过这次分解，使所得的三个关系均满足每一个非主属性均完全函数依赖于码(sno，cno)的要求，即在每一个关系中，不存在任何一个非主属性部分函数依赖于码的情况，因此，这三个关系均达到第二范式，记作：

Student∈2NF

Course∈2NF

S_C∈2NF

定义 1.10　若R ∈ 1NF，且每一个非主属性完全函数依赖于码，则 R∈2NF。

通俗地讲，2NF 要求表中的行不能重复，必须是唯一的，即消除关系的函数依赖中存在的部分函数依赖。

继续查看这三个关系是否存在问题。分析发现关系学生 Student、关系课程 Course 已经不存在插入异常、修改异常和删除异常，但有少量数据冗余的存在；而关系选修 S_C 中这四个问题仍然存在，究其原因是因为在该关系中依然存在有传递函数依赖，因此，分解关系选修 S_C 得到关系选修 SC 和关系教师 Teacher，两个关系分别表示为：

SC(sno，cno，grade，tno)

Teacher(tno，tname)

通过这次分解，使所得的两个关系均满足每一个非主属性都不传递依赖于码(sno，cno)的要求，因此，这两个关系均达到第三范式，记作：

SC ∈ 3NF

Teacher ∈ 3NF

回看关系学生 Student、关系课程 Course 同样也不存在非主属性对码的传递依赖，因此，同样也均达到第三范式。

定义 1.11　关系模式 R<U，F>中若不存在这样的码 X，属性组 Y 及非主属性 Z（Z ⊈ Y）使得X → Y，Y → Z成立，Y ↛ X，则R < U，F >∈ 3NF。

3NF 要求一个数据库表中不包含已在其他表中存在的非主码信息。

再看一个例子：假设有关系仓库管理 Storehouse_Manage(sid，pid，mid，count)，sid 代表仓库编号、pid 代表存放产品编号、mid 代表管理员编号、count 代表存放数量；且一个管理员只在一个仓库工作；一个仓库可以存储多种商品。该关系中存在的函数依赖有：

(sid，pid) →(mid，count)

(mid，pid) → (sid，count)

所以，(sid，pid)和(mid，pid)都是 StorehouseManage 的候选码，表中的唯一非候选码为存放数量 count，它是符合第三范式的。但是，由于存在如下决定关系：

sid→mid

mid→sid

仓库编号 sid 是决定因素，但其不包含候选码。

同样的，管理员编号 mid 也是决定因素，但不包含候选码。

按照 BCNF 对“关系中每个决定因素均是候选码”的要求，所以该关系不满足 BCNF。如果一个关系只有一个候选码，并满足 3NF，则该关系必定满足 BCNF，并且其 3NF 与 BCNF 完全相同。

对关系仓库管理 Storehouse_Manage 进行分解得到两个新关系分别为关系仓库管理员 S_M 和关系仓库产品 S_P，两个关系分别表示为：

S_M(sid，mid)

S_P(sid，pid，count)

定义 1.12 关系模式 $R<U，F>\in 1NF$。若 $X\rightarrow Y$ 且 $Y\notin X$ 时 X 必含有码，则 $R<U，F>\in BCNF$。

由 BCNF 的定义可以得到：BCNF 不仅排除了任何属性（包括主属性和非主属性）对码的部分依赖和传递依赖，而且排除了主属性之间的传递依赖和部分依赖。

3NF 和 BCNF 是在函数依赖的条件下对模式分解所能达到的最大程度。一个模式中的关系模式如果都属于 BCNF，那么在函数依赖范围内，它已经实现了彻底的分离，已消除了插入和删除的异常。3NF 的“不彻底”性表现在可能存在主属性对码的部分依赖和传递依赖。

总结，从 1NF 规范到 2NF，必须消除非主属性对码的部分依赖；从 2NF 规范到 3NF 的，必须消除非主属性对码的传递依赖；从 3NF 规范到 BCNF，必须消除主属性对码的部分和传递依赖。

关系规范化就是将关系按照一定的原则不断地分解为多个关系的过程，通过分解使关系逐步达到较高范式，任何一个非规范化的关系经过分解都可以达到 3NF。

关系分解的两个基本原则是：

（1） 关系分解后必须可以无损连接。无损即分解后信息不能丢失，连接即对分解后的关系进行某种连接运算后能使之到分解前的状态。

（2） 分解后的关系要相互独立。分解后的各个关系之间应是相互独立的，如果修改某一个关系不会涉及到修改另外一个关系。

习题

1. 填空题。

（1） 在关系模式 R(A,B,C,D)中，存在函数依赖关系{A→B,A→C,A→D,(B,C)→A}，则候选码是_____，关系模式 R(A,B,C,D)属于_____NF。

（2）在关系模式 R(D,E,G)中，存在函数依赖关系{E→D,(D,G)→E}，则候选码是_____，关系模式 R(D,E,G)属于_____NF。

（3）在关系模式 R(A,C,D)中，存在函数依赖关系{A→C,A→D}，则候选码是_____，关系模式 R(A,C,D)最高可以达到_____ NF。

2. 选择题。

（1） 学生表(id,name,sex,age,depart_id,depart_name)，存在的函数依赖是 id→{name,sex,age,depart_id}；dept_id→dept_name，其满足（　　）。

A．1NF　　B．2NF　　C．3NF　　D．BCNF

（2） 设有关系模式 R(S,D,M)，其函数依赖集 F＝{S→D,D→M}，则关系模式 R 的规范化程度最高达到（　　）。

A．1NF　　B．2NF　　C．3NF　　D．BCNF

（3） 设有关系模式 R(A,B,C,D)，其数据依赖集 F＝{(A,B)→C,B→D}，则关系模式 R 的规范化程度最高达到（　　）。

A．1NF　　B．2NF　　C．3NF　　D．BCNF

（4） 已知关系模式 R(A,B,C,D,E)及其上的函数依赖集合 F＝{A→D,B→C,E→A}，则该关系模式的候选码是（　　）。

A．(A,B)　　B．(B,E)　　C．(C,D)　　D．(D,E)

（5） 已知关系模式 R(A,B,C,D,E)及其上的函数相关性集合，F={A→B,E→A,(C,E)→D}，该关系的候选码是（　　）。

A．AE　　B．BE　　C．CE　　D．DE

（6） 设有关系模式 R(A,B,C,D)及其上的函数相关性集合 F＝{B→A,(B,C)→D},那么关系 R 最高是（　　）。

A．第一范式　　B．第二范式　　C．第三范式　　D．BCNF 范式

（7）在关系模式 R(A,B,C)中，函数依赖集 F={(A,B)→C,B→C}，则 R 最高达到(　　)。

A．1NF　　B．2NF　　C．3NF　　D．BCNF

3. 下表给出一数据集，请判断它是否可直接作为关系数据库中的关系，若不行，则改造成为尽可能好的并能作为关系数据库中关系的形式，同时说明进行这种改造的理由。

系　名	课 程 名	教 师 名
计算机系	DB	李军，刘强
机械系	CAD	金山，宋海
造船系	CAM	王华
自控系	CTY	张红，曾健

4. 设有如下所示的关系 R。

职 工 号	职 工 名	年　龄	性　别	部 门 号	部 门 名
20070711	张广智	29	男	001	财务部
20070731	李丽	32	女	002	采购部
20070722	王坤	46	男	001	财务部
20070756	刘凯	38	男	003	保卫部

试问 R 是否属于 3NF？为什么？若不是，它属于第几范式？并把它规范化为 3NF？

5. 设有关系模式 R(A,B,C)，数据依赖集 F={AB→C,C→A}，R 属于第几范式？

6. 设有关系模式 R(U,F)，其中：

U＝{A,B,C,D,E}，F＝{A→D,E→D,D→B,BC→D,DC→A}

求出 R 的候选码。

7. 设有关系模式R(A,B,C)，数据依赖集F={AB→C,C→A}，R属于第几范式？为什么？

8. 指出下列关系模式是第几范式?并说明理由。

（1） R(X,Y,Z)　　F＝{XY→Z}

（2） R(x,Y,z)　　F＝{Y→Z,XZ→Y}

（3） R(X,Y,Z)　　F＝{Y→Z,Y→X,X→YZ}

（4） R(x,Y,z)　　F＝{X→Y,X→Z}

（5） R(W,X,Y,Z)　　F＝{X→Z,WX→Y}

任务4　数据库设计

数据库设计是建立数据库及其应用系统的技术，是信息系统开发和建设中的核心技术，具体说，数据库设计是指利用现有的数据库管理系统，针对具体的应用要求构建适合的数据库模式，建立数据库及其应用的系统，使之能有效地收集、存储、操作和管理数据，满足各类用户的信息与处理要求。数据库设计问题是数据库在应用领域的主要研究课题。在数据库领域内，常常把使用数据库的各类系统统称为数据库应用系统。

1. 数据库设计概述

数据库设计虽然是一项应用课题，但它涉及的内容很广泛，所以设计一个性能良好的数据库并不容易。数据库设计的质量与设计者的知识、经验和水平有密切的关系。数据库设计中面临的主要困难和问题有：

① 懂得计算机与数据库的人一般都缺乏所开发数据库应用系统领域的业务知识和实际经验，而熟悉该业务且有经验的人通常又不懂得计算机软件开发，同时具备这两方面知识的人少之又少；

② 在开发初期往往不能明确应用业务所需要数据库系统的目标；

③ 客户的需求往往在一开始不很明确，而是在设计过程中不断修改或提出新的要求，甚至在数据库建立之后还会要求修改数据库结构和增加新的应用；

④ 缺乏很完善的设计工具和方法；

⑤ 数据库应用系统业务千差万别，很难找到一种适合所有应用业务的工具和方法，这就增加了研究数据库自动生成工具的难度，因此，开发适合一切数据库应用系统业务的全自动数据库生成工具是不可能的。

在进行数据库设计时，必须确定系统的目标，这样可以确保开发工作进展顺利，并能提高工作效率，保证数据模型的准确和完整。数据库设计的最终目标是数据库必须能够满足客户对数据的存储和处理需求，同时定义系统的长期和短期目标，能够提高系统的服务以及新数据库的性能期望值——客户对数据库的期望也是非常重要的。新的数据库能在多大程度上方便终端用户？新数据库的近期和长期发展计划是什么？是否所有的手工处理过程都可以自动实现？现有的自动化处理是否可以改善？这些都只是定义一个新的数据库设计目标时所必须考虑的一部分问题或因素。

成功的数据库系统应具备如下一些特点：

（1） 功能强大。

① 能准确地表示业务数据；

② 使用方便，易于维护；
③ 对最终用户操作的响应时间合理；
④ 便于数据库结构的改进；
⑤ 便于数据的检索和修改；
⑥ 维护数据库的工作较少；
⑦ 有效的安全机制以确保数据安全；
⑧ 冗余数据最少或不存在；
⑨ 便于数据的备份和恢复；
⑩ 数据库结构对终端用户来说是透明的。

数据库设计的工作量大且比较复杂，是一项数据库工程，也是一项软件工程。数据库设计的很多阶段都可以对应于软件工程的阶段，软件工程的某些方法和工具同样也适合于数据库工程，但由于数据库设计是与用户的业务需求紧密相关的，因此，它还有很多自己的特点。

（2） 综合性。

数据库设计涉及的范围很广，包含了计算机专业知识及业务系统的专业知识，同时它还要解决技术及非技术两方面的问题。

非技术问题包括组织机构的调整，经营方针的改变，管理体制的变更等。这些问题都不是设计人员所能解决的，但新的管理信息系统要求必须有与这相适应的新的组织机构、新的经营方针、新的管理体制，这就是一个较为尖锐的矛盾。另一方面，鉴于同时具备数据库和业务两方面知识的人很少，因此，数据库设计者一般都需要花费相当多的时间去熟悉应用业务系统知识，这一过程有时很麻烦，可能会使设计人员产生厌烦情绪，从而影响系统的最后成功。而且，由于承担部门和应用部门是一种委托雇佣关系，在客观上存在着一种对立，当在某些问题上意见不一致时会使双方关系比较紧张，这在 MIS（管理信息系统）中尤为突出。

（3） 结构设计与行为设计相分离。

结构设计是指数据库的模式结构设计，包括概念结构、逻辑结构和存储结构；行为设计是指应用程序设计，包括功能组织、流程控制等方面的设计。在传统的软件工程中，比较注重处理过程的设计，不太注重数据结构的设计。在一般的应用程序设计中只要可能就会推迟数据结构的设计，这种方法对于数据库设计就不太适用。

数据库设计与传统的软件工程的做法正好相反。数据库设计的主要精力首先是放在数据结构的设计上，比如数据库的表结构、视图等。

为了使数据库设计更合理、更有效，需要有效的指导原则，这种原则就称为数据库设计方法。

首先，一个好的数据库设计方法，应该能在合理的期限内，以合理的工作量，产生一个有实用价值的数据库结构。这里的“实用价值”是指满足用户关于功能、性能、安全性、完整性及发展需求等方面的要求，同时又服从特定 DBMS 的约束，可以用简单的数据模型来表达。其次，数据库设计方法还应具有足够的灵活性和通用性，不但能够使具有不同经验的人使用，而且不受数据模型及 DBMS 的限制。最后，数据库设计方法应该是可再生的，即不同的设计者使用同一方法设计同一问题时，可以得到相同或相似的设计结果。

多年来，经过人们不断的努力和探索，提出了各种数据库设计方法。运用软件工程的思想和方法提出的各种设计准则和规程都属于规范设计方法。

新奥尔良（New Orleans）方法是一种比较著名的数据库设计方法，这种方法将数据库设计分为四个阶段：需求分析、概念结构设计、逻辑结构设计和物理结构设计，如图 1-36 所示。这种方法注重数据库的结构设计，而不太考虑数据库的行为设计。S.B.Yao 等人又将数据库设计分为五个阶段，主张数据库设计应包括设计系统开发的全过程，并在每一阶段结束时进行评审，以便及早发现设计错误，及早纠正。各阶段也不是严格线性的，而是采取“反复探寻、逐步求精”的方法。在设计时从数据库应用系统设计和开发的全过程来考察数据库设计问题，既包括数据库模型的设计，也包括围绕数据库展开的应用处理的设计。在设计过程努力把数据库设计和系统其他部分的设计紧密结合，把数据和处理的需求、分析、抽象、设计和实现在各个阶段同时进行，相互参照，相互补充，以完善两方面的设计。

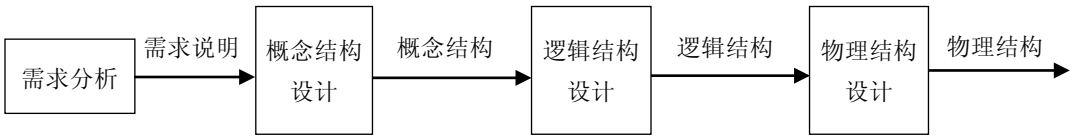


图 1-36　新奥尔良方法的数据库设计步骤

常见的数据库设计方法有：基于 E-R 模型的数据库设计方法，基于第三范式的设计方法，基于抽象语法规范的设计方法等都是在数据库设计的不同阶段上使用的具体技术和方法。数据库设计方法从本质上看仍然是手工设计方法，其基本思想是过程迭代和逐步求精。

按照规范设计的方法，同时考虑数据库及其应用系统开发的全过程，将数据库设计分为如下几个阶段：

① 需求分析；
② 概念结构设计；
③ 逻辑结构设计；
④ 物理结构设计；
⑤ 数据库实施；
⑥ 数据库运行和维护。

数据库设计全过程如图 1-37 所示。

数据库设计开始之前，首先必须选定参加设计的人员，包括系统分析人员、数据库设计人员和程序员、用户和数据库管理员。系统分析和数据库设计人员是数据库设计的核心人员，他们将自始至终参与数据库设计，他们的水平决定了数据库应用系统的质量。用户和数据库管理员在数据库设计中也是举足轻重的，他们主要参加需求分析和数据库的运行维护，他们的积极参与不但能加速数据库设计，而且也是决定数据库设计的质量的重要因素。程序员则在系统实施阶段参与进来，分别负责编制程序和准备软硬件环境。

如果所设计的数据库应用系统比较复杂，还应该考虑是否需要使用数据库设计工具和 CASE 工具以提高数据库设计质量并减少设计工作量。

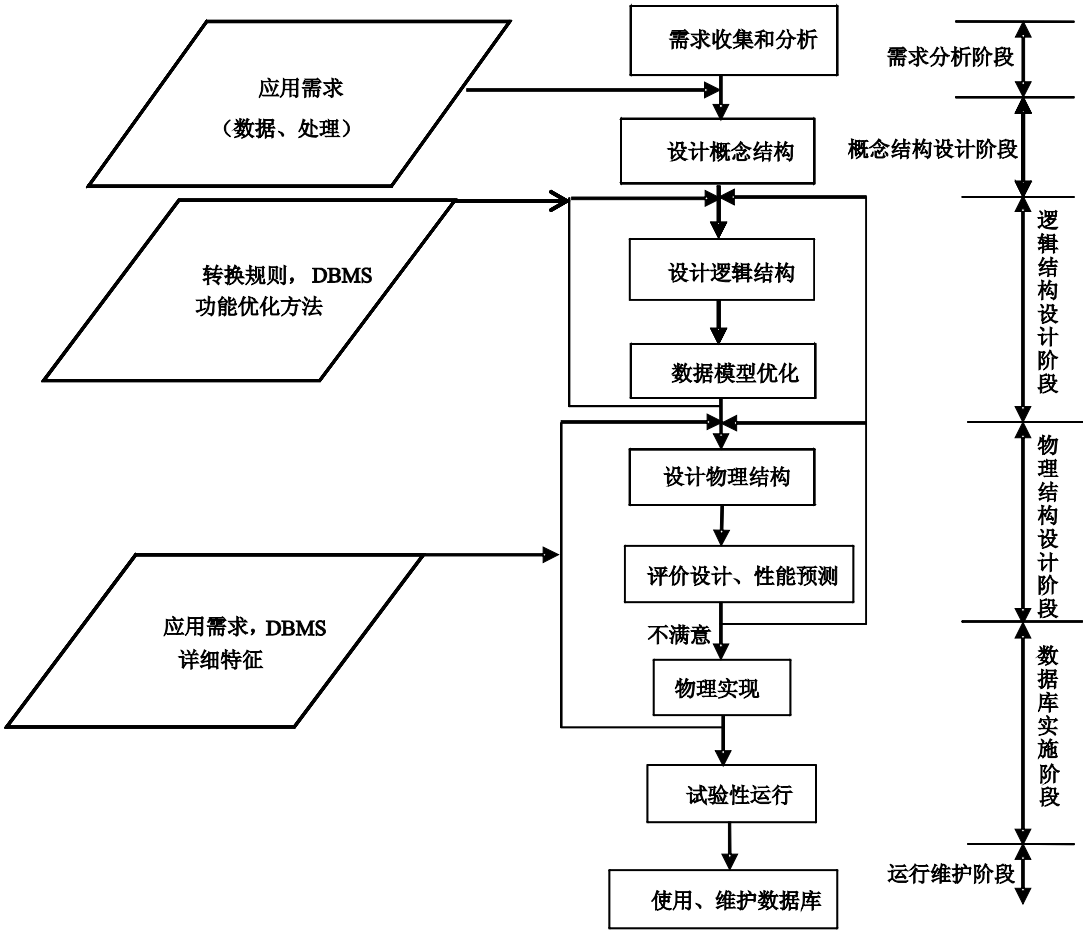


图 1-37　数据库设计的全过程

（1）　需求分析阶段。

进行数据库设计首先必须准确了解与分析用户需求（包括数据与处理）。需求分析是整个设计过程的基础，是最困难、最耗费时间的一步。作为地基的需求分析是否做得充分与准确，决定了在其上构建数据库大厦的速度与质量。需求分析做得不好，甚至会导致整个数据库设计返工重做。

（2）　概念结构设计阶段。

概念结构设计是整个数据库设计的关键，它通过对用户需求进行综合、归纳与抽象，形成一个独立于具体 DBMS 的概念模型。

（3）　逻辑结构设计阶段。

逻辑结构设计是将概念结构转换为某个 DBMS 所支持的数据模型，并对其进行优化。

（4）　数据库物理设计阶段。

数据库物理设计是为逻辑数据模型选取一个最适合应用环境的物理结构（包括存储结构和存取方法）。

（5） 数据库实施阶段。

在数据库实施阶段，设计人员运用 DBMS 提供的数据语言及其宿主语言，根据逻辑设计和物理设计的结果建立数据库，编制与调试应用程序，组织数据入库，并进行试运行。

（6） 数据库运行和维护阶段。

数据库应用系统经过试运行后即可投入正式运行。在数据库系统运行过程中必须不断地对其进行评价、调整与修改。

设计一个完善的数据库应用系统是不可能一蹴而就的，它往往是上述六个阶段的不断反复。

需要指出的是，这个设计步骤既是数据库设计的过程，也包括了数据库应用系统的设计过程。在设计过程中把数据库的设计和对数据库中数据处理的设计紧密结合起来，将这两个方面的需求分析、抽象、设计、实现在各个阶段同时进行，相互参照，相互补充，以完善两方面的设计。事实上，如果不了解应用环境对数据的处理要求，或没有考虑如何去实现这些处理要求，是不可能设计一个良好的数据库结构的。

2. 需求分析

需求分析简单地说就是分析用户的要求。需求分析是设计数据库的起点，需求分析的结果是否准确地反映了用户的实际要求，将直接影响到后面各个阶段的设计，并影响到设计结果是否合理和实用。

1） 需求分析的任务。

需求分析的任务是通过详细调查现实世界要处理的对象（组织、部门、企业等），充分了解原系统（手工系统或计算机系统）工作概况，明确用户的各种需求，确定新系统功能的同时，收集支持系统目标的基础数据及其处理方法。需求分析是在用户调查的基础上，通过分析，逐步明确用户对系统的需求，包括数据需求和围绕这些数据的业务处理需求。

调查的重点是“数据”和“处理”，通过调查要从用户那里获得对数据库的下列要求：

（1） 信息需求。定义所设计数据库系统用到的所有信息，明确用户将向数据库中输入什么样的数据，从数据库中要求获得哪些内容，将要输出哪些信息，也就是明确在数据库中需要存储哪些数据，对这些数据将做哪些处理等，同时还要描述数据间的联系等。

（2） 处理需求。定义系统数据处理的操作功能，描述操作的次序，包括操作的执行频率和场合，操作与数据间的联系，还要明确用户要完成哪些处理功能，每种处理的执行频度，用户需求的响应时间以及处理的方式（比如是联机处理还是批处理）等。

（3） 安全性与完整性要求。安全性要求描述系统中不同用户对数据库的使用和操作情况，完整性要求描述数据之间的关联关系以及数据的取值范围要求。

在需求分析中，通过自顶向下、逐步分解的方法分析系统，任何一个系统都可以抽象为数据流图的形式。

数据流图是从“数据”和“处理”两方面表达数据处理的一种图形化表示方法。在需求分析阶段，不必确定数据的具体存储方式，这些问题留待进行物理结构设计时考虑。数据流图中的“处理”抽象地表达了系统的功能需求，系统的整体功能要求可以分解为系统的若干子功能要求，通过逐步分解的方法，可以将系统的工作过程细分，直至表达清楚为止。

需求分析是整个数据库设计中最重要的一步，是其他步骤的基础。如果把整个数据库设计当成一个系统工程看待，那么需求分析就是这个系统工程的最原始的输入信息。如果这一步做得不好，那么后续的设计即使再优化也只能前功尽弃，所以这一步特别重要。

需求分析也是最困难、最麻烦的一步，其困难之处不在于技术上，而在于要了解、分析、表达客观世界并非易事，这也是数据库自动生成工具的研究中最困难的部分。目前，许多自动生成工具都绕过这一步，先假定需求分析已经有结果，这些自动工具就以这一结果作为后续步骤的输入。

2）需求分析的方法。

需求分析首先要调查清楚用户的实际需求，与用户达成共识，然后再分析这些需求。调查用户需求的重点是“数据”和“处理”，为达到这一目的，在调查前要拟定调查提纲。调查时要抓住两个“流”，即“信息流”和“处理流”，而且调查中要不断地将这两个“流”结合起来。调查的任务是调研现行系统的业务活动规则，并提取描述系统业务的现实系统模型。

通常情况下，调查用户的需求包括三方面内容，即系统的业务现状、信息源流及外部要求。

（1）业务现状。

业务现状包括：业务方针政策、系统的组织机构、业务内容、约束条件和各种业务的全过程。

（2）信息源流。

信息源流包括：各种数据的种类、类型及数据量，各种数据的源头、注射和终点，各种数据的产生、修改、查询及更新过程和频率以及各种数据与业务处理的关系。

（3）外部要求。

外部要求包括：对数据保密性的要求，对数据完整性的要求，对查询响应时间的要求，对新系统使用方式的要求，对输入方式的要求，对输出报表的要求，对各种数据精度的要求，对吞吐量的要求，对未来功能、性能及应用范围扩展的要求。

在进行需求调查时，实际上就是发现现行业务系统的动作事实。常用的发现事实的方法有检查文档、面谈、观察操作中的业务、研究和问卷调查等。

① 检查文档。

当要深入了解为什么客户需要数据库应用时，检查用户的文档是非常有用的。检查文档有助于提供与问题相关的业务信息，如果问题与现在系统相关，则一定有与该系统相关的文档。检查与目前系统相关的文档、表格、报告和文件是一种非常好的快速理解系统的方法。

② 面谈。

面谈是最常用的，通常也是最有用的事实发现方法，通过面对面谈话获取有用信息。面谈还有其他用处，比如找出事实，确认、澄清事实，得到所有最终用户，标识需求，集中意见和观点。但是，使用面谈这种技术需要良好的交流能力，面谈的成功与否依赖于谈话者的交流技巧，而且，面谈也有它的缺点，比如非常消耗时间，为了保证面谈成功，必须选择合适的谈话人选，准备的问题涉及范围要广，要引导谈话有效地进行。

③ 观察业务的运转。

观察是用来理解一个系统的最有效的事实发现方法之一。使用这个技术可以参与或者观察做事的人以了解系统。当用其他方法调研的数据的有效性值得怀疑或者系统特定方面的复杂性阻碍了最终用户做出清晰的解释时，这种技术尤其有用。与其他事发现相比，成功的观察要求做非常多的准备。为了确保成功，要尽可能多地了解要观察的人和活动，例如，所观察的活动的低谷、正常以及高峰期分别是什么时候？

④ 研究。

研究是通过计算机行业的杂志、参考书和因特网等来查找是否有类似的解决此问题的方法，甚至可以查找和研究是否存在解决此问题的软件包。但这种方法也有很多缺点，比如，如果存在解决此问题的方法，则可以节省很多时间，但如果没有，则可能会非常浪费时间。

⑤ 问卷调查。

另一种事实发现方法是通过问卷来调查。问卷是一种有着特定目的的小册子，这样可以在控制答案的同时，集中一大群人的意见。当和大批用户打交道，其他的事实发现技术都不能有效地把这些事实列成表格时，就可以采用问卷调查的方式，问卷有两种格式：自由形式问卷和固定形式。

在自由格式问卷上，答卷人提供答案有更大的自由。问题提出后，答卷人在题目后的空白地方写答案。例如：“你当前收到的是什么报表，它们有什么用”，“这些报告是否存在问题？如果有，请说明”。自由格式问卷存在的问题是答卷人的答案可能难以列成表格，而且，有时答卷人可能答非所问。

在固定格式问卷上，包含的问题答案是特定的。给定一个问题，回答者必须从提供的答案中选择一个。因此，结果容易列表。但另一方面，答卷人不能提供一些有用的附加信息，如现在的业务系统的报告形式非常理想，不必改动。答卷人可以选择的答案有“是”或者“否”，或者一组选项，包括“非常赞同”、“同意”、“没意见”、“不同意”和“强烈反对”等。

做需求调查时，往往需要同时采用上述多种方法，但无论使用哪种调查方法，都必须有用户的积极参与和配合。

3） 需求分析的步骤。

需求分析的步骤如下：

（1） 分析用户活动，产生用户活动图。

这一步要了解用户当前的业务活动和职能，分析其处理过程。如果一个业务流程比较复杂，要把它分解为几个子处理，使每个处理功能明确、界面清楚，分析之后画出用户活动图（即用户的业务流程图）。本部分参考软件工程相关材料。

（2） 确定系统范围，产生系统范围图。

这一步是确定系统的边界。在和用户经过充分讨论的基础上，确定计算机所能进行的数据处理和范围，确定哪些工作由人工完成，哪些工作由计算机系统完成，即确定人机界面。本部分参考软件工程相关材料。

（3） 分析用户活动所涉及的数据，产生数据流图。

在这一过程中，要深入分析用户的业务处理过程，以数据流图的形式表示出数据的流

动和对数据所做的加工。

数据流图（Data Flow Diagram，DFD）有 4 个基本成分：数据流（用箭头表示）、加工或处理（用圆圈表示）、文件（用右边无边框矩形或双线段表示）和外部实体（数据流的源点和终点，用长方形表示。图 1-38 是学生住宿管理数据流图。

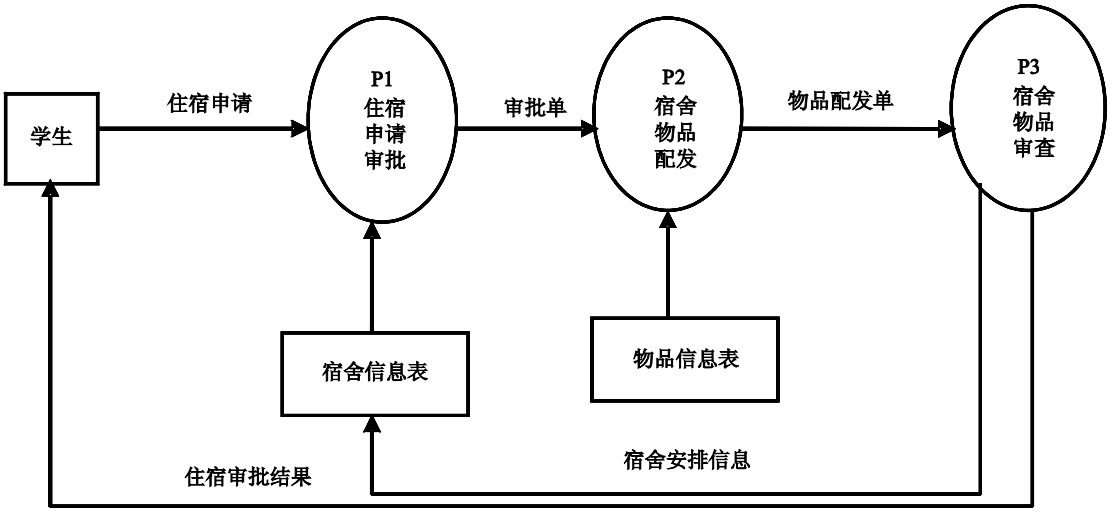


图 1-38　学生住宿管理数据流图

（4） 分析系统数据，产生数据字典。

数据流图表达了数据和处理的关系，是从动态的角度描述系统；而数据字典是系统中各类数据描述的集合，是进行详细的数据收集和数据分析所获得的主要成果，是从静态的角度描述系统。

数据字典通常包括数据项、数据结构、数据流、数据存储和处理过程五个部分。其中数据项是数据的最小组成单位，若干个数据项可以组成一个数据结构，数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容。

① 数据项。

数据项是不可再分的数据单位。对数据项的描述通常包括以下内容：

数据项描述＝｛数据项名，数据项含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系，数据项之间的联系｝

其中“取值范围”、“与其他数据项的逻辑关系”（例如该数据项等于另几个数据项的和，该数据项值等于另一数据项的值等）定义了数据的完整性约束条件，是设计数据检验功能的依据。

可以用关系规范化理论为指导，用数据依赖的概念分析和表示数据项之间的联系，即按实际语义，写出每个数据项之间的数据依赖，它们是数据库逻辑设计阶段数据模型优化的依据。

下面以“学号”为例介绍数据项在数据字典中的描述：

数据项名：学号

含义说明：唯一标识每位学生

别名：学生编号

类型：字符型

长度：10

取值范围：0000000000～9999999999

取值含义：前两位是学生入学年份的后两位，第三、四位是学生所在院系的编号，第五、六位是学生所学专业编号，第七、八位是学生所在班级编号，最后两位是该生在该班级的编号。

② 数据结构。

数据结构反映了数据之间的组合关系。一个数据结构可以由若干数据项组成，也可以由若干数据结构组成，或由若干数据项和数据结构混合组成。对数据结构的描述通常包括以下内容：

数据结构描述＝｛数据结构名，含义说明，组成：｛数据项或数据结构｝｝

下面以“宿舍”为例，介绍数据结构在数据字典中的描述：

数据结构名：宿舍

含义说明：是学生宿舍管理系统的核心数据结构，定义了一个宿舍的有关信息

组成：宿舍编号+所在楼层+宿舍编号+床位编号+可住人数+已入住人数+宿舍电话+住宿费+椅子数+桌子数+备注

③ 数据流。

数据流是数据结构在系统内传输的路径。对数据流的描述通常包括以下内容：

数据流描述＝｛数据流名，说明，数据流来源，数据流去向，组成：｛数据结构｝，平均流量，高峰期流量｝

其中，“数据流来源”是说明该数据流来自哪个过程。“数据流去向”是说明该数据流将到哪个过程去。“平均流量”是指在单位时间（每天、每周、每月等）里的传输次数。“高峰期流量”则是指在高峰时期的数据流量。

下面以“物品配发单”为例，介绍数据流在数据字典中的描述：

数据流名：物品配发单

说明：为即将入住学生配发的物品信息

数据流去向：物品审核

组成：{学生信息}+{物品信息}+{宿舍信息}+{审报时间}

平均流量：500 次/月

高峰期流量：3000 次/天

④ 数据存储。

数据存储是数据结构停留或保存的地方，也是数据流的来源和去向之一。它可以是手工文档或手工凭单，也可以是计算机文档。对数据存储的描述通常包括以下内容：

数据存储描述＝｛数据存储名，说明，编号，输入的数据流，输出的数据流，组成：｛数据结构，数据量，存取频度，存取方式｝｝

其中“存取频度”指每小时或每天或每周存取几次、每次存取多少数据等信息。“存取方式”包括是批处理还是联机处理；是检索还是更新；是顺序检索还是随机检索等。另外，“输入的数据流”要指出其来源，“输出的数据流”要指出其去向。

下面以“宿舍信息表”为例，介绍数据存储在数据字典中的描述：

数据存储：宿舍信息表

说明：记录学校宿舍使用情况

编号：01

输入的数据流：学生信息+宿舍信息+配发物品信息

输出的数据流：宿舍安排信息

组成：学号+宿舍号+床位号+桌子数+椅子数+入住时间+住宿时长+是否延期

数据量：每年 5500 条

存取频度：每天 100 条

存取方式：随机存取

⑤ 处理过程。

处理过程的具体处理逻辑一般用判定表或判定树来描述。数据字典中只需要描述处理过程的说明性信息，通常包括以下内容：

处理过程描述＝｛处理过程名，说明，输入：｛数据流｝，输出：｛数据流｝，处理：｛简要说明｝｝

其中“简要说明”中主要说明该处理过程的功能及处理要求。功能是指该处理过程用来做什么（而不是怎么做），处理要求包括处理频度要求，如单位时间里处理多少事务、多少数据量、响应时间要求等。这些处理要求是后面物理设计的输入及性能评价的标准。

下面以“住宿审批”为例，介绍处理过程在数据字典中的描述：

处理过程名：住宿审批

说明：为申请入住的学生分配宿舍

输入：学生申请表、宿舍信息

输出：审批单

处理：在学生提出申请后，为学生分配宿舍，要求同一间宿舍只能安排同一性别的学生，同一个学生只能安排在一个宿舍中，每个学生的居住面积不小于 3 平方米，安排新生宿舍其处理时间应不超过 10 分钟

可见，数据字典是关于数据库中数据的描述，即元数据，而不是数据本身。数据字典是在需求分析阶段建立，在数据库设计过程中不断修改、充实、完善的。明确地把需求收集和分析作为数据库设计的第一阶段是十分重要的。这一阶段收集到的基础数据（用数据字典来表达）和一组数据流程图（Data Flow Diagram，简称 DFD）是下一步进行概念设计的基础。

在调查了解了用户需求以后，还需要进一步分析和表达用户的需求。在众多的分析方法中结构化分析方法（Structured Analysis，简称 SA 方法）是一种简单实用的方法。SA 方法从最上层的系统组织机构入手，采用自顶向下、逐层分解的方式分析系统。在 SA 方法中，系统中的数据则借助数据字典（Data Dictionary，DD）来描述，处理过程的处理逻辑常常借助判定表或者判定树来描述。

最后，要强调两点：

（1） 需求分析阶段的一个重要而困难的任务是收集将来应用所涉及的数据，设计人员应充分考虑到可能的扩充和改变，使设计易于更改，系统易于扩充。

（2） 必须强调用户的参与，这是数据库应用系统设计的特点。数据库应用系统和广泛的用户有密切的联系，许多人要使用数据库，数据库的设计和建立又可能对更多人的工作环境产生重要影响。因此用户的参与是数据库设计不可分割的一部分。在数据分析阶段，任何调查研究没有用户的积极参加是寸步难行的。设计人员应该和用户取得共同的语言，帮助不熟悉计算机的用户建立数据库环境下的共同概念，并对设计工作的最后结果承担共同的责任。

3. 概念结构设计

将需求分析得到的用户需求抽象为信息结构即概念模型的过程就是概念结构设计。它是整个数据库设计的关键。

1） 概念结构。

在需求分析阶段所得到的应用需求应该首先抽象为信息世界的结构，才能更好地、更准确地用某一 DBMS 实现这些需求。

概念结构的主要特点是：

（1） 能真实、充分地反映现实世界，包括事物和事物之间的联系，能满足用户对数据的处理要求，是对现实世界的一个真实模型。

（2） 易于理解，从而可以用它和不熟悉计算机的用户交换意见，用户的积极参与是数据库的设计成功的关键。

（3） 易于更改，当应用环境和应用要求改变时，容易对概念模型修改和扩充。

（4） 易于向关系、网状、层次等各种数据模型转换。

概念结构是各种数据模型的共同基础，它比数据模型更独立于机器、更抽象，从而更加稳定。

描述概念模型的有力工具是 E-R 模型。有关 E-R 模型的基本概念已在任务 1 中介绍。

2） 概念结构设计的方法与步骤。

设计概念结构通常有四类方法：

（1） 自顶向下。即首先定义全局概念结构的框架，然后逐步细化。

（2） 自底向上。即首先定义各局部应用的概念结构，然后将它们集成起来，得到全局概念结构。

（3） 逐步扩张。首先定义最重要的核心概念结构，然后向外扩充，以滚雪球的方式逐步生成其他概念结构，直至总体概念结构。

（4） 混合策略。即将自顶向下和自底向上相结合，用自顶向下策略设计一个全局概念结构的框架，以它为骨架集成由自底向上策略中设计的各局部概念结构。

其中，最经常采用的策略是自底向上方法，即自顶向下地进行需求分析，然后再自底向上地设计概念结构。

这里只介绍自底向上设计概念结构的方法。它通常分为两步：第 1 步是抽象数据并设计局部视图，第 2 步是集成局部视图，得到全局的概念结构。

3） 数据抽象与局部视图设计。

概念结构是对现实世界的一种抽象。所谓抽象是对实际的人、物、事和概念进行人为处理，抽取所关心的共同特性，忽略非本质的细节，并把这些特性用各种概念精确地加以

描述，这些概念组成了某种模型。

4）　视图的集成。

各子系统的分 E-R 图设计好以后，下一步就是要将所有的分 E-R 图综合成一个系统的总 E-R 图，即

合并分 E-R 图，生成初步 E-R 图。

消除不必要的冗余，设计基本 E-R 图。

下面将用 E-R 模型来描述学生住宿管理系统的概念结构。

（1）　确定实体。

学生住宿管理系统中实体有学生、宿舍楼、宿舍、宿舍物品、宿舍管理人员。

（2）　给出各实体的属性。

学生住宿管理系统中各实体的属性如下：

学生的属性有：学号、院系、姓名、性别、年龄、电话、家庭住址、家庭联系人姓名、家庭联系人电话。

宿舍楼的属性有：宿舍楼编号、所在校区、宿舍楼位置、宿舍间数、建造开始时间、建造完工时间。

宿舍的属性有：宿舍号、宿舍类型、收费标准、标配电量、标本水量、可入住人数、已入住人数。

宿舍物品的属性有：物品编号、物品名称、物品价格。

宿舍管理人员的属性有：工号、姓名、性别、年龄、工龄、工种、入职时间。

（3）　定义实体之间的联系。

一个宿舍中可以住多名学生，每名学生只能住在一个宿舍中，因此，宿舍与学生之间是一对多的联系。

一栋宿舍楼由多个宿舍组成，因此，宿舍楼与宿舍之间是一对多的联系。

一栋宿舍楼有一名宿舍管理人员，一名宿舍管理人员只能在一栋宿舍楼中工作，因此，宿舍楼与宿舍管理人员之间是一对一的联系。

一栋宿舍楼配有多种宿舍物品，每种宿舍物品配发到各个宿舍中，因此，宿舍楼与宿舍物品之间是多对多的联系。

每个学生配备有多种宿舍物品，每种宿舍物品可以配发给多名学生，因此，学生与宿舍物品之间是多对多的联系。

（4）　绘制学生宿舍管理系统的 E-R 图如图 1-39 所示。

在以上学生实体与宿舍管理人员实体的属性命名中，出现重名不同义情况，因此，对重复的名字进行重命名以消除此现象。

实体学生的属性重命名后为：

学号，院系，学生姓名，学生性别，学生年龄，电话，家庭住址，家庭联系人姓名，家庭联系人电话。

宿舍管理人员实体的属性重命名后为：

宿舍管理人员的属性有：工号、工人姓名、工人性别、工人年龄、工龄、工种、入职时间。

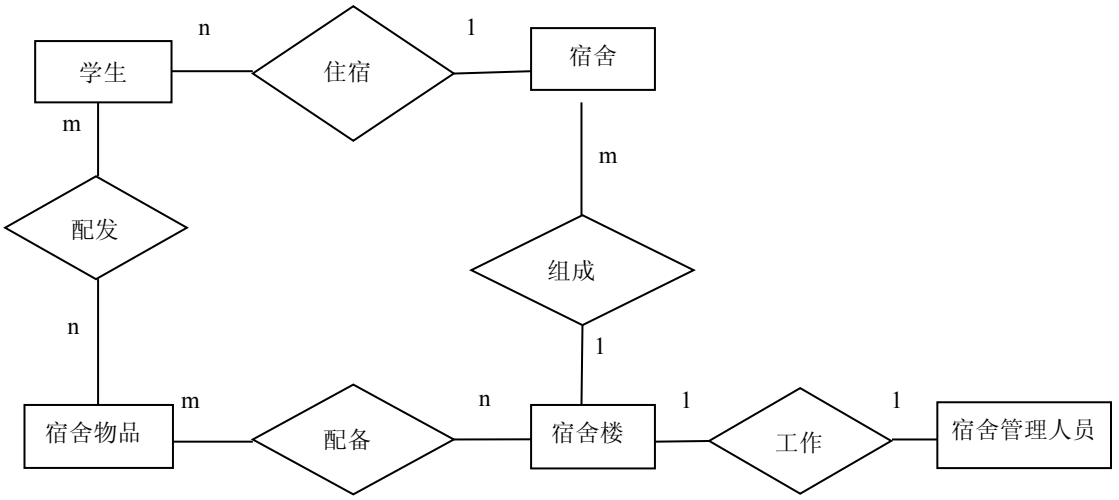


图 1-39　学生宿舍管理系统的 E-R 图

4. 逻辑结构设计

概念结构是独立于任何一种数据模型的信息结构。逻辑结构设计的任务就是把概念结构设计阶段设计好的基本 E-R 图转换为与选用 DBMS 产品所支持的数据模型相符合的逻辑结构。

从理论上讲，设计逻辑结构应该选择最适于相应概念结构的数据模型，然后对支持这种数据模型的各种 DBMS 进行比较，从中选出最合适的 DBMS。但实际情况往往是已给定了某种 DBMS，设计人员没有选择的余地。目前 DBMS 产品一般支持关系、网状、层次三种模型中的某一种，对某一种数据模型，各个机器系统又有许多不同的限制，提供不同的环境与工具。所以设计逻辑结构时一般要分三步进行：

（1） 将概念结构转换为一般的关系、网状、层次模型；

（2） 将转换来的关系、网状、层次模型向特定 DBMS 支持下的数据模型转换；

（3） 对数据模型进行优化。

1） E-R 图向关系模型的转换。

E-R 图向关系模型的转换要解决的问题是如何将实体和实体间的联系转换为关系模式，如何确定这些关系模式的属性和码。

关系模型的逻辑结构是一组关系模式的集合。E-R 图则是由实体、实体的属性和实体之间的联系三个要素组成的。所以将 E-R 图转换为关系模型实际上就是要将实体、实体的属性和实体之间的联系转换为关系模式，这种转换一般遵循如下原则：

（1） 一个实体型转换为一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。

（2） 对于实体间的联系则有以下不同的情况：

① 一个 1:1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，每个实体的码均是该关系的候选码。如果与某一端实体对应的关

系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。

② 一个 1:n 联系可以转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为 n 端实体的码。

③ 一个 m:n 联系转换为一个关系模式。与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为各实体码的组合。

④ 三个或三个以上实体间的一个多元联系可以转换为一个关系模式。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为各实体码的组合。

⑤ 具有相同码的关系模式可合并。

2） 形成了一般的数据模型后，下一步就是向特定的 RDBMS 的模型转换。设计人员必须熟悉所用 RDBMS 的功能与限制。这一步是依赖于机器的，不能给出一个普遍的规则，但对于关系模型来说，这种转换通常都比较简单，不会有太多的困难。

依据上述理论，将学生住宿管理系统的 E-R 图转换出相应的关系模式如下：

学生关系（学号，院系，学生姓名，学生性别，学生年龄，电话，家庭住址，家庭联系人姓名，家庭联系人电话，宿舍楼编号，宿舍号）

宿舍楼关系（宿舍楼编号，所在校区，宿舍楼位置，宿舍间数，建造开始时间，建造完工时间）

宿舍关系（宿舍号，宿舍类型，收费标准，标配电量，标配水量，可入住人数，已入住人数，宿舍楼编号）

宿舍物品关系（物品编号，物品名称，物品价格）

宿舍管理人员关系（工号，工人姓名，工人性别，工人年龄，工龄，工种，入职时间，宿舍楼编号）

宿舍物品发放关系（学号，物品编号，领取数量，领取日期，领取人）

宿舍物品配备关系（宿舍楼编号，物品编号，配备数量，发放数量，剩余数量）

3） 数据模型的优化。

数据库逻辑设计的结果不是唯一的。为了进一步提高数据库应用系统的性能，还应该根据应用需要适当地修改、调整数据模型的结构，这就是数据模型的优化。关系数据模型的优化通常以规范化理论为指导，方法为：

（1） 确定数据依赖。根据需求分析阶段所得到的语义，分别写出每个关系模式内部各属性之间的数据依赖以及不同关系模式属性之间的数据依赖。

（2） 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系，具体方法已在关系数据理论中讲解。

（3） 按照数据依赖的理论对关系模式逐一进行分析，考察是否存在部分函数依赖、传递函数依赖、多值依赖等，确定各关系模式分别属于第几范式。

（4） 按照需求分析阶段得到的处理要求，分析对于这样的应用环境这些模式是否合适，确定是否要对某些模式进行合并或分解。

必须注意的是，并不是规范化程度越高的关系就越优。例如，当查询经常涉及到两个或多个关系模式的属性时，系统经常进行连接运算。连接运算的代价是相当高的，可以说

关系模型低效的主要原因就是连接运算引起的。这时可以考虑将这几个关系合并为一个关系。因此在这种情况下，第二范式甚至第一范式也许是合适的。

又如，非 BCNF 的关系模式虽然从理论上分析会存在不同程度的更新异常或冗余，但如果在实际应用中对此关系模式只是查询，并不执行更新操作，则就不会产生实际影响。所以对于一个具体应用来说，到底规范化到什么程度，需要权衡响应时间和潜在问题两者的利弊决定。

规范化理论为数据库设计人员判断关系模式优劣提供了理论标准，可用来预测模式可能出现的问题，使数据库设计工作有了严格的理论基础。

依据关系数据理论，对学生宿舍管理系统中每个关系进行分析。

关系学生的函数依赖是：

$$\text{学号} \xrightarrow{F} (\text{院系，学生姓名，学生性别，学生年龄，电话，家庭住址，家庭联系人姓名，家庭联系人电话，宿舍楼编号，宿舍号})$$

关系宿舍楼的函数依赖是：

$$\text{宿舍楼编号} \xrightarrow{F} (\text{所在校区，宿舍楼位置，宿舍间数，建造开始时间，建造完工时间})$$

关系宿舍的函数依赖是：

$$\text{宿舍号} \xrightarrow{F} (\text{宿舍类型，收费标准，标配电量，标配水量，可入住人数，已入住人数，宿舍楼编号})$$

关系宿舍物品的函数依赖是：

$$\text{物品编号} \xrightarrow{F} \text{物品名称，物品价格}$$

关系宿舍管理人员的函数依赖是：

$$\text{工号} \xrightarrow{F} (\text{工人姓名，工人性别，工人年龄，工龄，工种，入职时间，宿舍楼编号})$$

关系宿舍物品发放的函数依赖是：

$$(\text{学号，物品编号}) \xrightarrow{F} (\text{领取数量，领取日期，领取人})$$

关系宿舍物品配备的函数依赖是：

$$(\text{宿舍楼编号，物品编号}) \xrightarrow{F} (\text{配备数量，发放数量，剩余数量})$$

从各个关系模式的数据依赖看出，对于学生宿舍管理系统的每个关系模式来说，均达到 3NF 的要求，因此，适于该系统的使用。

（5） 对关系模式进行必要的分解，提高数据操作的效率和存储空间的利用率。常用的两种分解方法是水平分解和垂直分解。

（6） 设计用户子模式。

将概念模型转换为全局逻辑模型后，还应该根据局部应用需求，结合具体 DBMS 的特点，设计用户的外模式。

目前，关系数据库管理系统一般都提供视图（View）概念，可以利用这一功能设计更符合局部用户需要的用户外模式。

定义数据库全局模式主要从系统的时间效率、空间效率、易维护等角度出发。由于用户外模式与模式是相对独立的，因此在定义用户外模式时可以注重考虑用户的习惯与方便，包括：

① 使用更符合用户习惯的别名。

在合并各分 E-R 图时，曾做了消除命名冲突的工作，以使数据库系统中同一关系和属性具有唯一的名字。这在设计数据库整体结构时是非常必要的。用 View 机制可以在设计用户 View 时重新定义某些属性名，使其与用户习惯一致，以方便使用。

② 可以对不同级别的用户定义不同的 View，以保证系统的安全性。

假设对于学生宿舍管理系统中的关系模式学生（学号，院系，学生姓名，学生性别，学生年龄，电话，家庭住址，家庭联系人姓名，家庭联系人电话，宿舍楼编号，宿舍号），可以在学生关系上建立两个视图，以方便使用。

为学生建立视图：

个人信息视图（学号，姓名，院系，电话，宿舍楼编号，宿舍号）

为宿舍管理人员建立视图：

学生信息（学号，姓名，电话，院系，家庭住址，家庭联系人姓名，家庭联系人电话）

个人信息视图中只包含允许学生查询的属性，而为宿舍管理人员建立的学生信息视图中包含允许宿舍管理人员查询的属性。这样就可以防止用户非法访问本来不允许他们查询的数据，保证了系统的安全性。

③ 简化用户对系统的使用。

如果某些局部应用中经常要使用某些很复杂的查询，为了方便用户，可以将这些复杂查询定义为视图，用户每次只对定义好的视图进行查询，大大简化了用户的使用。

5. 数据库的物理设计

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于给定的计算机系统。为一个给定的逻辑数据模型选取一个最适合应用要求的物理结构的过程，就是数据库的物理设计。

数据库的物理设计通常分为两步：

（1） 确定数据库的物理结构，在关系数据库中主要指存取方法和存储结构；

（2） 对物理结构进行评价，评价的重点是时间和空间效率。

如果评价结果满足原设计要求，则可进入到物理实施阶段，否则，就需要重新设计或修改物理结构，有时甚至要返回逻辑设计阶段修改数据模型。

不同的数据库产品所提供的物理环境、存取方法和存储结构有很大差别，能供设计人员使用的设计变量、参数范围也很不相同，因此没有通用的物理设计方法可遵循，只能给出一般的设计内容和原则。希望设计优化的物理数据库结构，使得在数据库上运行的各种事务响应时间小、存储空间利用率高、事务吞吐率大。为此首先对要运行的事务进行详细分析，获得选择物理数据库设计所需要的参数。其次，要充分了解所用 RDBMS 的内部特征，特别是系统提供的存取方法和存储结构。

对于数据库查询事务，需要得到如下信息：

（1） 查询的关系；

（2） 查询条件所涉及的属性；

（3） 连接条件所涉及的属性；

（4） 查询的投影属性。

对于数据更新事务，需要得到如下信息：

（1） 被更新的关系；

（2） 每个关系上的更新操作条件所涉及的属性；

（3） 修改操作要改变的属性值。

除此之外，还需要知道每个事务在各关系上运行的频率和性能要求。例如，事务 T 必须在 10 秒钟内结束，这对于存取方法的选择具有重大影响。

上述这些信息是确定关系的存取方法的依据。

应注意的是，数据库上运行的事务会不断变化、增加或减少，以后需要根据上述设计信息的变化调整数据库的物理结构。

通常对于关系数据库物理设计的内容主要包括 ：

（1） 为关系模式选择存取方法；

（2） 设计关系、索引等数据库文件的物理存储结构。

下面就介绍这些设计内容和方法。

1） 关系模式存取方法选择。

数据库系统是多用户共享的系统，对同一个关系要建立多条存取路径才能满足多用户的多种应用要求。物理设计的任务之一就是要确定选择哪些存取方法，即建立哪些存取路径。

存取方法是快速存取数据库中数据的技术。数据库管理系统一般都提供多种存取方法。常用的存取方法有三类。第一类是索引方法，目前主要是 B+树索引方法；第二类是聚簇（Cluster）方法；第三类是 HASH 方法。

B+树索引方法是数据库中经典的存取方法，使用最普遍。

2） 确定数据库的存储结构。

确定数据库物理结构主要指确定数据的存放位置和存储结构，包括确定关系、索引、聚簇、日志、备份等的存储安排和存储结构；确定系统配置等。

确定数据的存放位置和存储结构要综合考虑存取时间、存储空间利用率和维护代价三方面的因素。这三个方面常常是相互矛盾的，因此需要进行权衡，选择一个折中方案。

（1） 确定数据的存放位置。

为了提高系统性能，应该根据应用情况将数据的易变部分与稳定部分、经常存取部分和存取频率较低部分分开存放。

例如，目前许多计算机都有多个磁盘，因此可以将表和索引放在不同的磁盘上，在查询时，由于两个磁盘驱动器并行工作，可以提高物理 I/O 读写的效率；也可以将比较大的表分放在两个磁盘上，以加快存取速度，这在多用户环境下特别有效；还可以将日志文件与数据库对象（表、索引等）放在不同的磁盘上以改进系统的性能。此外，数据库的数据备份和日志文件备份等只在故障恢复时才使用，而且数据量很大，可以存放在磁带上。

由于各个系统所能提供的对数据进行物理安排的手段、方法差异很大，因此设计人员应仔细了解给定的 RDBMS 提供的方法和参数，针对应用环境的要求，对数据进行适当的物理安排。

（2） 确定系统配置。

DBMS 产品一般都提供了一些系统配置变量、存储分配参数，供设计人员和 DBA 对数据库进行物理优化。初始情况下，系统都为这些变量赋予合理的默认值。但是这些值不一定适合每一种应用环境，在进行物理设计时，需要重新对这些变量赋值，以改善系统的性能。

系统配置变量很多，例如：同时使用数据库的用户数，同时打开的数据库对象数，内存分配参数，缓冲区分配参数（使用的缓冲区长度、个数），存储分配参数，物理块的大小，物理块装填因子，时间片大小，数据库的大小，锁的数目等。这些参数值影响存取时间和存储空间的分配，在物理设计时就要根据应用环境确定这些参数值，以使系统性能最佳。

在物理设计时对系统配置变量的调整只是初步的，在系统运行时还要根据系统实际运行情况做进一步的调整，以期切实改进系统性能。

3） 评价物理结构。

数据库物理设计过程中需要对时间效率、空间效率、维护代价和各种用户要求进行权衡，其结果可以产生多种方案，数据库设计人员必须对这些方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构。

评价物理数据库的方法完全依赖于所选用的 DBMS，主要是从定量估算各种方案的存储空间、存取时间和维护代价入手，对估算结果进行权衡、比较，选择出一个较优的合理的物理结构。如果该结构不符合用户需求，则需要修改设计。

6. 数据库的实施与维护

完成数据库的物理设计之后，设计人员就要用 RDBMS 提供的数据定义语言和其他实用程序将数据库逻辑设计和物理设计结果严格描述出来，成为 DBMS 可以接受的源代码，再经过调试产生目标模式，然后就可以组织数据入库了，这就是数据库实施阶段。

（1） 数据的载入和应用程序的调试。

数据库实施阶段包括两项重要的工作，一项是数据的载入，另一项是应用程序的编码和调试。

一般数据库系统中，数据量都很大，而且数据来源于部门中的各个不同的单位，数据的组织方式、结构和格式都与新设计的数据库系统有相当的差距，组织数据录入就要将各类源数据从各个局部应用中抽取出来，输入计算机，再分类转换，最后综合成符合新设计的数据库结构的形式，输入数据库。因此这样的数据转换、组织入库的工作是相当费力费时的工作。特别是原系统是手工数据处理系统时，各类数据分散在各种不同的原始表格、凭证、单据之中。在向新的数据库系统中输入数据时，还要处理大量的纸质文件，工作量就更大。

由于各个不同的应用环境差异很大，不可能有通用的转换器，DBMS 产品也不提供通用的转换工具。为提高数据输入工作的效率和质量，应该针对具体的应用环境设计一个数据录入子系统，由计算机来完成数据入库的任务。

由于要入库的数据在原来的系统中的格式结构与新系统中不完全一样，有的差别可能还比较大，不仅向计算机内输入数据时发生错误，转换过程中也有可能出错。因此在原数据入库之前要采用多种方法对它们进行检验，以防止不正确的数据入库，这部分的工作在整个数据输入子系统中是非常重要的。

在设计数据输入子系统时还要注意原有系统的特点，例如对原有系统是人工数据处理系统的情况，尽管新系统的数据结构可能与原系统有很大差别，在设计数据输入子系统时，尽量让输入格式与原系统结构相近，这不仅使处理手工文件比较方便，更重要的是减少用户出错的可能性，保证数据输入的质量。现有的 DBMS 一般都提供不同 DBMS 之间数据转换的工具，若原来是数据库系统，就可以利用新系统的数据转换工具，先将原系统中的表转换成新系统中相同结构的临时表，再将这些表中的数据分类、转换、综合成符合新系统的数据模式，插入相应的表中。

数据库应用程序的设计应该与数据库设计同时进行，因此在组织数据入库的同时还要调试应用程序。应用程序的设计、编码和调试的方法、步骤在软件工程等课程中有详细讲解，这里就不赘述了。

（2） 数据库的试运行。

在原有系统的数据有一小部分已输入数据库后，就可以开始对数据库系统进行联合调试，这又称为数据库的试运行。

这一阶段要实际运行数据库应用程序，执行对数据库的各种操作，测试应用程序的功能是否满足设计要求。如果不满足，对应用程序部分则要修改、调整，直到达到设计要求为止。

在数据库试运行时，还要测试系统的性能指标，分析其是否达到设计目标。在对数据库进行物理设计时已初步确定了系统的物理参数值，但一般的情况下，设计时的考虑在许多方面只是近似的估计，和实际系统运行总有一定的差距，因此必须在试运行阶段实际测量和评价系统性能指标。事实上，有些参数的最佳值往往是经过运行调试后找到的。如果测试的结果与设计目标不符，则要返回物理设计阶段，重新调整物理结构，修改系统参数，某些情况下甚至要返回逻辑设计阶段，修改逻辑结构。

这里特别要强调两点：

第一，上面已经讲到组织数据入库是十分费时费力的事，如果试运行后还要修改数据库的设计，还要重新组织数据入库。因此应分期分批地组织数据入库，先输入小批量数据做调试用，待试运行基本合格后，再大批量输入数据，逐步增加数据量，逐步完成运行评价。

第二，在数据库试运行阶段，由于系统还不稳定，硬、软件故障随时都可能发生。而系统的操作人员对新系统还不熟悉，误操作也不可避免，因此应首先调试运行 DBMS 的恢复功能，做好数据库的转储和恢复工作。一旦故障发生，能使数据库尽快恢复，尽量减少对数据库的破坏。

（3） 数据库的运行和维护。

数据库试运行合格后，数据库开发工作就基本完成，即可投入正式运行了。但是，由于应用环境在不断变化，数据库运行过程中物理存储也会不断变化，对数据库设计进行评价、调整、修改等维护工作是一个长期的任务，也是设计工作的继续和提高。

在数据库运行阶段，对数据库经常性的维护工作主要是由 DBA 完成的，它包括：

① 数据库的转储和恢复。

数据库的转储和恢复是系统正式运行后最重要的维护工作之一。DBA 要针对不同的应用要求制定不同的转储计划，以保证一旦发生故障能尽快将数据库恢复到某种一致的状态，并尽可能减少对数据库的破坏。

② 数据库的安全性、完整性控制。

在数据库运行过程中，由于应用环境的变化，对安全性的要求也会发生变化，比如有的数据原来是机密的，现在是可以公开查询的了，而新加入的数据又可能是机密的了。系统中用户的密级也会改变。这些都需要 DBA 根据实际情况修改原有的安全性控制。同样，数据库的完整性约束条件也会变化，也需要 DBA 不断修正，以满足用户要求。

③ 数据库性能的监督、分析和改造。

在数据库运行过程中，监督系统运行，对监测数据进行分析，找出改进系统性能的方法是 DBA 的又一重要任务。目前有些 DBMS 产品提供了监测系统性能参数的工具，DBA 可以利用这些工具方便地得到系统运行过程中一系列性能参数的值。DBA 应仔细分析这些数据，判断当前系统运行状况是否最佳，应当做哪些改进。例如，调整系统物理参数，或对数据库进行重组织或重构造等。

④ 数据库的重组织与重构造。

数据库运行一段时间后，由于记录不断增、删、改，会使数据库的物理存储情况变坏，降低了数据的存取效率，数据库性能下降，这时 DBA 就要对数据库进行重组织，或部分重组织（只对频繁增、删的表进行重组织）。DBMS 一般都提供数据重组织用的实用程序。在重组织的过程中，按原设计要求重新安排存储位置、回收垃圾、减少指针链等，提高系统性能。

数据库的重组织，并不修改原设计的逻辑和物理结构，而数据库的重构造则不同，它是指部分修改数据库的模式和内模式。

由于数据库应用环境发生变化，增加了新的应用或新的实体，取消了某些应用，有的实体与实体间的联系也发生了变化等，使原有的数据库设计不能满足新的需求，需要调整数据库的模式和内模式。例如，在表中增加或删除某些数据项，改变数据项的类型，增加或删除某个表，改变数据库的容量，增加或删除某些索引等。当然数据库的重构也是有限的，只能做部分修改。如果应用变化太大，重构也无济于事，说明此数据库应用系统的生命周期已经结束，应该设计新的数据库应用系统了。

习题

1. 某医院病房计算机管理中心需要如下信息：

科室：科名、科地址、科电话、医生姓名

病房：病房号、床位号、所属科室名

医生：姓名、职称、所属科室名、年龄、工作证号

病人：病历号、姓名、性别、诊断、主管医生、病房号

其中，一个科室有多个病房、多个医生，一个病房只能属于一个科室，一个医生只属于一个科室，但可负责多个病人的诊治，一个病人的主管医生只有一个。

请设计该计算机管理系统的 E-R 模型。

2. 一个图书馆借阅管理数据库要求提供下述服务：

（1） 可随时查询书库中现有书籍的品种、数量与存放位置。所有各类书籍均可由书号唯一标识。

（2） 可随时查询书籍借还情况，包括借书人单位、姓名、借书证号、借书日期和还书日期。我们约定：任何人可借多种书，任何一种书可为多个人所借，借书证号具有唯一性。

（3） 当需要时，可通过数据库中保存的出版社的电报编号、电话、邮编及地址等信息向相应出版社增购有关书籍。我们约定，一个出版社可出版多种书籍，同一本书仅为一个出版社出版，出版社名具有唯一性。

请根据以上情况和假设设计满足需求的 E-R 模型。

3. 假设要根据黄淮学院中各院系、专业、学生、班级、社团等信息建立一个数据库，一个院系有若干专业，每个专业只隶属于一个院系；每个专业每年只招一个班，每个班只属于一个专业；每个班有若干名学生，每名学生只属于一个班。每个学生可以参加多个社团，每个社团有若干学生，学生参加某社团有加入年份。根据描述完成黄淮学院学生管理系统数据库设计要求：

（1） 请为该系统设计合适的 E-R 模型。

（2） 请给出该系统的关系模型，指出关系的码和外码。

任务 5　数据库的完整性约束机制

数据库完整性（Database Integrity）是指数据库中数据在逻辑上的一致性、正确性、有效性和相容性。关系数据模型中有四类完整性约束：实体完整性、参照完整性、域完整性和用户定义的完整性，其中实体完整性和参照完整性约束条件，称为关系的两个不变性。用户定义的完整性包括非空约束、检查约束和唯一值约束三种。对于较复杂的约束条件，通常利用触发器实现。下面以实例的形式给出几种完整性约束机制在 SQL Server 2008 下的实现过程。

1. 主码

【例 1】创建学生 student 表（结构同前），学号 sid 定义为主码。

```
CREATE TABLE student(
sid CHAR(10) PRIMARY  KEY ,/*在列级定义主码*/
sPassword varchar(20),
sName  varchar(10),
sClassName varchar(30),
Ssex  CHAR(2) ,
sBirthday datetime,
sPolitical varchar(10),
fresh bit,
sPhotoPath varchar(255),
sEmail varchar(30)
);
```

【例 2】创建成绩表 score（结构同前），学号 sid 和课程号 cid 定义为主码。

```
CREATE  TABLE  score(
sid char(10),
cid varchar(10),
academic year varchar(20),
semester varchar(10),
usalResults int,
examResults int,
experimentalResults int,
totalScore int,
PRIMARY KEY (sid,cid)/*表级定义主码*/
);
```

2. 外码

【例 1】重建主码部分【例 2】中成绩表 score，添加 sid 参照 student 表的主码 sid 及 cid 参照 course 表的主码 Cno。

```
CREATE TABLE score(
sid char(10),
cid varchar(10),
academic_year varchar(20),
semester varchar(10),
usalResults int,
examResults int,
experimentalResults int,
totalScore int,
FOREIGN KEY (sid) REFERENCES student(sid),/*在表级定义参照完整性*/
FOREIGN KEY (cid) REFERENCES course(cid),/*在表级定义参照完整性*/
PRIMARY KEY (sid,cid)/*表级定义主码*/
);
```

【例 2】重建【例 1】中成绩表 score，添加当从学生表 student 中删除一条学生信息时，级连删除该生的所有选课信息，当修改学生表 student 中学号 sid 时，级连更新修改选课表中所有该生学号。

```
CREATE TABLE score(
sid char(10),
cid varchar(10),
academic_year varchar(20),
semester varchar(10),
usalResults int,
examResults int,
experimentalResults int,
totalScore int,
FOREIGN KEY (sid) REFERENCES student(sid)/*在表级定义参照完整性*/
ON  DELETE  CASCADE--当删除 Student 表中的元组时，级连删除 SC 表中相应的元组*/
ON  UPDATE  CASCADE,--当更新 Student 表中的 Sno 时，级连更新 SC 表中相应的元组*/
FOREIGN KEY (cid) REFERENCES course(cid),/*在表级定义参照完整性*/
```

```
PRIMARY KEY (sid,cid)/*表级定义主码*/
);
```

3. 非空约束

【例】重建主码部分【例 1】中学生表 student，约定密码和姓名不允许取空值。

```
CREATE TABLE student(
sid CHAR(10) PRIMARY  KEY ,/*在列级定义主码*/
sPassword varchar(20) not null,
sName  varchar(10)not null,
sClassName varchar(30),
Ssex  CHAR(2) ,
sBirthday datetime,
sPolitical varchar(10),
fresh bit,
sPhotoPath varchar(255),
sEmail varchar(30)
);
```

4. 唯一值约束

【例】重建非空约束中学生表 student，约定电子邮箱取唯一值。

```
CREATE TABLE student(
sid CHAR(10) PRIMARY  KEY ,/*在列级定义主码*/
sPassword varchar(20) not null,
sName  varchar(10)not null,
sClassName varchar(30),
Ssex  CHAR(2) ,
sBirthday datetime,
sPolitical varchar(10),
fresh bit,
sPhotoPath varchar(255),
sEmail varchar(30) UNIQUE
);
```

5. 默认值约束

【例】重建唯一值约束中学生表 student，约定若不输入学生性别，默认取“男”，若不输入是否为应届生，默认取值为“1”，即是应届生。

```
CREATE TABLE student(
sid CHAR(10) PRIMARY  KEY ,/*在列级定义主码*/
sPassword varchar(20) not null,
sName  varchar(10)not null,
sClassName varchar(30),
Ssex  CHAR(2) default '男',
sBirthday datetime,
sPolitical varchar(10),
fresh bit default 1,
```

```
    sPhotoPath varchar(255),
    sEmail varchar(30) UNIQUE
    );
```

6. 检查约束

【例 1】重建唯一值约束中学生表 student，约定学生性别只能取“男”或者“女”。

```
    CREATE TABLE student(
    sid CHAR(10) PRIMARY KEY,
    sPassword varchar(20) not null,
    sName  varchar(10)not null,
    sClassName varchar(30),
    Ssex  CHAR(2) default '男' CHECK (sSex IN('男','女')),
    sBirthday datetime,
    sPolitical varchar(10),
    fresh bit default 1,
    sPhotoPath varchar(255),
    sEmail varchar(30) UNIQUE
    );
```

【例 2】重建主码例中成绩表 score，约定成绩应该均在[0，100]之间。

```
    CREATE TABLE score(
    sid char(10),
    cid varchar(10),
    academic_year varchar(20),
    semester varchar(10),
    usalResults int CHECK (usalResults between 0 and 100),
    examResults int CHECK (examResults>=0 AND examResults<=100),
   experimentalResults int CHECK (experimentalResults>=0  AND  experimentalResults
<=100),
    totalScore int CHECK (totalScore>=0 AND totalScore<=100),
    FOREIGN KEY (sid) REFERENCES student(sid)/*在表级定义参照完整性*/
    ON  DELETE  CASCADE--当删除 Student 表中的元组时，级连删除 SC 表中相应的元组*/
    ON  UPDATE  CASCADE,--当更新 Student 表中的 Sno 时，级连更新 SC 表中相应的元组*/
    FOREIGN KEY (cid) REFERENCES course(cid),/*在表级定义参照完整性*/
    PRIMARY KEY (sid,cid)/*表级定义主码*/
```

7. 完整性约束命名子句

完整性约束命名子句的格式是：

```
    CONSTRAINT  <完整性约束条件名>  [PRIMARY KEY 短语| FOREIGN KEY 短语| CHECK 短语]
```

【例 1】重建检查约束中例，为每一个约束添加名称。

```
    CREATE TABLE student(
    sid CHAR(10) CONSTRAINT c1 PRIMARY KEY,
    sPassword varchar(20) CONSTRAINT c2 not null,
    sName  varchar(10) CONSTRAINT c3 not null,
    sClassName varchar(30),
```

```
Ssex  CHAR(2) CONSTRAINT c4 default '男'
CONSTRAINT c5 CHECK (sSex IN('男','女')),
sBirthday datetime,
sPolitical varchar(10),
fresh bit CONSTRAINT c6 default 1,
sPhotoPath varchar(255),
sEmail varchar(30) CONSTRAINT c7 UNIQUE
);
```

对于表中的完整性限制，可以用 ALTER　TABLE 语句来修改表中的完整性限制。

【例 2】去掉学生表 student 中对性别的所有限制。

```
ALTER  TABLE  Student  DROP  CONSTRAINT  c4,c5 ;
```

【例 3】修改表 student 表中的约束条件，要求学号改为在 0000000000～9999990230 之间，即每年每个专业招两个班，每个班最多为 30 人。

先删除原来的约束，再添加新的约束。

```
ALTER TABLE student DROP CONSTRAINT c1 ;
ALTER TABLE Student
ADD CONSTRAINT c1 CHECK(sid  BETWEEN 0000000000 AND 9999990230) ;
```

8. 触发器

1） 触发器概念及类型。

触发器（Trigger）是创建在基本表或者视图上的一种特殊的存储过程。触发器的执行不是由程序调用，更不是手工启动，而是由事件来触发，当对一个表进行更新操作（insert，delete，update）时就会激活它执行，触发器通常应用在对复杂的数据关系进行强制约束，以达到数据的完整性约束和业务规则要求。

根据触发事件不同，将触发器分为 DML 触发器、DDL 触发器和登录触发器三种。

（1） DML（数据操纵语言 Data Manipulation Language）触发器：指触发器在数据库中发生 DML 事件时将启用。DML 事件即指在表或视图中修改数据的 insert、update、delete 语句。

（2） DDL（数据定义语言 Data Definition Language）触发器：指当服务器或数据库中发生 DDL 事件时将启用。DDL 事件即指在表或索引中的 create、alter、drop 语句。

（3） 登录触发器：指当用户登录 SQL Server 实例建立会话时触发。

其中 DML 触发器最为常用。

根据 DML 触发器触发的方式不同又分为以下两种情况：

① after 触发器：它是在执行 insert、update、delete 语句操作之后执行触发器操作。它主要是用于记录变更后的处理或检查，一旦发生错误，可以用 rollback transaction 语句来回滚本次事件，不过不能对视图定义 after 触发器。

② instead of 触发器：它在执行 insert、update、delete 语句操作之前执行触发器本身所定义的操作，并且 instead of 触发器可以定义在视图上，也可以定义在基本表上。

在 SQL Server 2008 中，DML 触发器的实现使用两个专用临时逻辑表 deleted 和 inserted。

这两个表的数据结构与触发器所在的表结构完全相同，它建立在数据库服务器的内存中，用户不能对它们进行修改，只能在触发器程序中查询表中数据，即用户只有只读的权限。当触发器执行完成后，它们会被自动删除。inserted 表用于存放用户在操作 insert、update、delete 语句后，更新的记录，比如插入一条数据，那么就会把这条记录插入到 inserted 表；deleted 表用于存放在操作 update、delete 语句前，创建触发器表中数据库，例如原来的表中有三条数据，那么在 deleted 表也有三条数据。也就是说，用户可以使用这两个临时的驻留内存的表，测试某些数据修改的效果及设置触发器操作的条件。

2）DML 触发器的主要作用及其优缺点。

SQL Server 数据库主要通过约束和触发器两种机制来实现数据完整性，约束机制只能对同一个表的列数据及表间的主码和外码数据进行限制，而触发器可以对表实现更为复杂的约束。DML 触发器的主要作用及其优缺点是：

（1）触发器可以实现数据库中的相关级联更新；

（2）DML 触发器可以防止恶意或者错误的 insert，update 以及 delete 操作，并强制执行比 check 约束定义的限制更为复杂的其他限制；

（3）触发器可以强制执行，与 check 约束不同，触发器可以引用其他表中的列；

（4）触发器也可以跟踪、监督数据库修改前后的状态，并根据其差异采取对策；

（5）一个表中的多个同类触发器 insert、update、delete 允许采取多个不同的对策，以响应同一个修改语句。

与此同时，触发器一定要慎用，虽然触发器功能强大，可以轻松可靠地实现许多复杂的功能，但由于触发器有“牵一发而动全身”的效果，如果过多触发器会造成数据库及应用程序的维护困难，同时对触发器过分的依赖，势必影响数据库的结构，同时增加了维护的复杂程度。

3）创建触发器的语法格式。

创建触发器的语法格式如下：

```
CREATE TRIGGER 触发器名
    ON { 表| 视图 }
    [ WITH ENCRYPTION ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ][, ] [ DELETE ] [, ] [ UPDATE ] }
    [ NOT FOR REPLICATION ]
    AS
    [ { IF UPDATE ( 列名 )
    [ { AND | OR } UPDATE ( 列名) ]
    [ ...n ]
    | IF ( COLUMNS_UPDATED ( ) updated_bitmask )
    column_bitmask [ ...n ]
    } ]
    SQL 语句
    }
```

参数说明：

（1）触发器名：创建触发器的名称。触发器命名遵照“见名知义”的原则，同时名

称必须符合标识符规则，并且在数据库中必须唯一，可以选择是否指定触发器所有者名称。

（2） 表 | 视图：是在其上执行触发器的表或视图，有时称为触发器表或触发器视图，可以选择是否指定表或视图的所有者名称。

（3） with encryption：对触发器的文本内容进行加密。使用 with encryption 可防止将触发器作为 SQL Server 复制的一部分发布。

（4） after：创建后触发器，即指定触发器只有在触发 SQL 语句中指定的所有操作都已成功执行后才激发。所有的引用级联操作和约束检查也必须成功完成后，才能执行此触发器。如果仅指定 for 关键字，则 after 是默认设置。注意：不能在视图上定义 after 触发器。

（5） instead of：创建替代触发器，即指定执行触发器而不是执行触发 SQL 语句，从而替代触发语句的操作。在基本表或视图上，每个 insert、update 或 delete 语句最多可以定义一个 instead of 触发器。然而，可以在每个具有 instead of 触发器的视图上定义视图。

instead of 触发器不能在 with check option 的可更新视图上定义。如果向指定了 with check option 选项的可更新视图添加 instead of 触发器，SQL Server 将产生一个错误。用户必须用 alter view 删除该选项后才能定义 instead of 触发器。

（6） { [delete] [，] [insert] [，] [update] }：是指定在基本表或视图上执行哪些数据修改语句时将激活触发器的关键字，必须至少指定一个选项。在触发器定义中允许使用以任意顺序组合的这些关键字。如果指定的选项多于一个，需用逗号分隔这些选项。

对于 instead of 触发器，不允许在具有 on delete 级联操作引用关系的表上使用 delete 选项。同样，也不允许在具有 on update 级联操作引用关系的表上使用 update 选项。

（7） not for replication：表示当复制进程更改触发器所涉及的表时，不应执行该触发器。

（8） as：是触发器要执行的操作。

（9） SQL 语句：是触发器的条件和操作。触发器条件指定其他准则，以确定 delete、insert 或 update 语句是否导致执行触发器操作。当尝试 delete、insert 或 update 操作时，Transact-SQL 语句中指定的触发器操作将生效。

触发器可以包含任意数量和种类的 Transact-SQL 语句。触发器旨在根据数据修改语句检查或更改数据；它不应将数据返回给用户。触发器中的 Transact-SQL 语句常常包含控制流语言。

（10）if update（列）：测试在指定的列上进行的 insert 或 update 操作，不可以用于 delete 操作，可以指定多列。因为在 on 子句中指定了表名，所以在 if update 子句中的列名前不要包含表名。

（11） if (columns_updated())测试是否插入或更新了提及的列，仅用于 insert 或 update 触发器中。

4） 触发器的维护。

（1） 修改触发器：与 alter table 类似，修改触发器的命令为 alter trigger。

（2） 启用或禁止触发器。

```
alter table <表名> disable trigger  <触发器名>表示禁用某个触发器
alter table <表名> enable trigger  <触发器名>表示启用某个触发器
alter table <表名> disable all triggers 表示禁用表上的所有的触发器
```

```
alter table <表名> enable all triggers 表示启用表上的所有的触发器
```

（3） 设置触发器的激发顺序。

如果在一个基本表上定义同一类型的多个后触发器，系统对这些触发器的触发顺序是随机的，但可以通过使用系统存储过程 sp_settriggerorder 约定第一个和最后一个后触发器。

```
sp_settriggerorder  triggername ,'first|last|none','insert|update|delete'
```

（4） 删除触发器。

```
delete trigger triggername
```

注意：删除触发器必须有相应的权限，没有权限不可以执行删除操作。

（5） 创建触发器。

【例 1】若删除一位学生信息时，检查该生是否选课，如果已经选课，则不允许删除该生信息。

```
use ssms
if(OBJECT ID('tri del student1') is not null)
drop trigger tri del student1
go
create trigger tri del student1
on student for delete
as
declare @sid old char(10),@sid count int
select @sid old=sid from deleted
select @sid count=count(*) from score where sid=@sid old
if(@sid count<>0)
begin
print '该生选的有课程，请您先删除所选课程，再删除该生信息！'
rollback transaction
end
else
begin
print '成功删除该生信息'
end
```

触发器测试示例代码如下：

```
insert  into  student(sid,sName,sPassword)  values('1334120112',' 刘 超 ',
'000000');
insert into score(sid,cid) values('1334120112','1')
insert into score(sid,cid) values('1334120112','2')
insert into score(sid,cid) values('1334120112','3')
delete from student where sid='1334120112'
select * from score;
```

【例 2】在成绩表 score 上，创建一个后触发触发器，当增加一个学生选课信息时，若不指定开课学年及开课学期，则自动确定开课年份、开课学期为课程信息表中确定的开课

学年和开课学期。

```
if(OBJECT_ID('tri_insert_score1') is not null)
drop trigger tri_insert_score1
go
create trigger tri_insert_score1
on score
for insert
as
begin
declare @sid char(10),@cid varchar(10),@year char(4),@term varchar(10)
select @sid=sid,@cid=cid,@year=academic_year,@term=semester from inserted
if(@year is null)
select @year=cOpenyear from course where cid=@cid;
if(@term is null)
select @term=cOpenterm from course where cid=@cid
delete from score where sid=@sid and cid=@cid
insert into score(sid,cid,academic_year,semester)
values(@sid,@cid,@year,@term)
end
```

触发器测试示例代码如下：

```
delete from score where sid='1434110101'
insert into score(sid,cid) values('1434110101','1');
select * from score;
```

【例 3】当修改一个学生学号时，级联对该生选课表中的学号进行修改。

```
if(object_id('tri_update_student2') is not null)
drop trigger tri_update_student2
go
create trigger tri_update_student2
on student
for update
as
if update(sid)
begin
declare @sid_new char(10),@sid_old char (10)
select @sid_new=sid from inserted
select @sid_old=sid from deleted
print @sid_old
print @sid_new
update score set sid=@sid_new where sid=@sid_old
end
```

触发器测试示例代码如下：

```
go
update student set sid='1334110121' where sid='1334110102'
select * from score
```

【例 4】当插入或修改一位学生信息时，对新的学号进行相容性检查，若学号中的院系编号在院系表 department 中不存在，则提示错误信息；否则，正确插入。

```
if(object_id('tri_insert_update_student3') is not null)
drop trigger tri_insert_update_student3
go
create trigger tri_insert_update_student3
on student
for insert,update
as
begin
declare @sno char(10),@dnonew char(2),@count int
select @sno=sid from inserted
set @dnonew=substring(@sno,3,2)
select @count=count(*) from department where did=@dnonew
print @count
if(@count!=1)
  begin
  print('该生学号中的院系编号不存在！')
  rollback
  end
else
  print('该生的输入信息基本正确并已经保存，谢谢您的使用！')
end
go
```

触发器测试示例代码如下：

```
sp_settriggerorder  'tri_insert_update_student3','first','update'
delete from student where sid='1534010101'
delete from student where sid='1514010101'
insert into student(sid,sName,sPassword)
values('1534010101','张三','000000');--院系编号为信息工程学院编号
insert into student(sid,sName,sPassword)
values('1514010101','李四','000000');--假设院系编号的院系不存在
```

习题

1. 基于成绩管理系统数据库 ssms，定义并验证如下约束：

（1） Student 表的 Ssex 列只能取“男”或“女”；

（2） Score 表的 totalScore 列的值在 0～100 之间；

（3） Student 表的 Sdept 列的值只能为两个字母；

（4） Score 表的 3 号课程的成绩不得高于 80；

（5） 要求 Student 表中，男生的年龄大于 18 岁，女生的年龄小于 18 岁；

（6） teacher 表的 tTitle（职称）值只能为：助教、讲师、教授、副教授，不能为工程师等；

（7） 在 SC 中不允许 1334120102 学生选修 5 号课程；

（8） teacher 表的 tTelephone 列只能取 11 位数字。

2. 创建一触发器，当对 score 表的 examScore 属性进行修改时，若分数增加了 10%，则将此次操作记录到另一个表 score_update(sid,cid,oldExamScore,newExamScore)中，其中 oldExamScore 是修改前的分数，newExamScore 是修改后的分数。

3. 创建一触发器，在学生（student）表增加一个学生记录时，对该生所在班级（class）的班级人数 cNumber 列的值加 1。

模块 2　开发环境的搭建与使用

学习目标

通过本模块的学习，熟练掌握 SQL Server 2008 R2 的安装和配置；熟悉 JDK 和 Tomcat 的安装及配置；掌握 Eclipse 的安装和使用；熟练掌握 SQL Server Management Studio 和 Eclipse 开发环境的使用；了解 SQL Server 系统数据库的作用和 SQL Server 2008 R2 的卸载。

主要内容

（1） SQL Server 2008 R2 的安装和配置。
（2） SQL Server 2008 R2 工具的使用。
（3） JDK 的安装与配置。
（4） Tomcat 的安装和配置。
（5） Eclipse 环境搭建及使用。

重点与难点

（1） SQL Server 2008 R2 的安装配置和使用。
（2） Tomcat 的安装。
（3） Eclipse 环境的使用。

案例 1　安装并熟悉SQL Server 2008

虽然 SQL Server 目前最新版本为 SQL Server 2014，但是，目前市场上还是使用 SQL Server 2008 版本的相对较多，因为 SQL Server 2008 相对比较稳定。下面就介绍在 Win7 环境下，SQL Server 2008 的安装和配置。

一、安装 SQL Server 2008 需要的软硬件环境

安装 SQL Server 2008 需要以下软件组件：.NET Framework 3.5 SP11 软件开发工具包（SDK），此开发包 Win7 系统中自带，无需安装和启动，自动工作。SQL Server 2008 对系统的要求如表 2-1 所示。

表 2-1　SQL Server 2008 系统需求

组　　件	要　　求
处理器	处理器类型：Pentium III 兼容处理器或速度更快的处理器 处理器速度：最低：1.0 GHz，建议：2.0 GHz 或更快的处理器
操作系统	Windows XP Professional SP2 Windows Server 2003 SP2 Windows Vista Windows Server 2008 Win7 Win8
内存	最小：1 GB 推荐：4 GB 或更大内存 最大：操作系统最大内存

二、SQL Server 2008 版本介绍

SQL Server 2008 分为服务器版和专业版两种，服务器版包括：数据中心版、企业版、标准版；专业版包括：开发版、工作组版、Web 版、简易版。不同版本的 SQL Server 能够满足单位和个人独特的性能、运行以及价格要求。表 2-2 是对 SQL Server 2008 各个服务器版本情况的介绍。

表 2-2　SQL Server 2008 服务器版包含的版本和定义

术　　语	定　　义
Datacenter（x86、x64 和 IA64）	SQL Server 2008 Datacenter 建立在 SQL Server 2008 Enterprise 基础之上，它提供了高性能的数据平台，这种平台可提供最高级别的可扩展性，以承载大量的应用程序工作负荷，支持虚拟化和合并，并管理组织的数据库基础结构，可帮助组织以经济高效的方式扩展其关键任务环境
Enterprise（x86、x64 和 IA64）	SQL Server 2008 Enterprise 提供了一个综合的数据平台，这一平台提供了内置安全性、可用性、可伸缩性并结合了稳定的商务智能功能
Standard（x86 和 x64）	SQL Server 2008 Standard 提供了一个全面的数据管理和商务智能平台，使小型组织能够顺利运行其应用程序，可帮助以最少的 IT 资源获得高效的数据库管理。SQL Server Standard for Small Business 包含 SQL Server Standard 的所有技术组件和功能，可以在拥有 75 台或更少计算机的小型企业环境中运行

SQL Server 专业版是针对特定的用户群体而设计的。表 2-3 介绍 SQL Server 2008 各个专业版。

表 2-3　SQL Server 2008 专业版包含的版本和定义

术　　语	定　　义
SQL Server Developer（x86、x64 和 IA64）	SQL Server Developer 支持开发人员构建基于 SQL Server 的任一种类型的应用程序。它包括 SQL Server Datacenter 的所有功能，但有许可限制，只能用作开发和测试系统，而不能用作生产服务器。SQL Server Developer 是构建和测试应用程序的人员的理想之选。可以升级 SQL Server Developer 以将其用于生产用途
SQL Server Workgroup（x86 和 x64）	SQL Server Workgroup 是运行分支位置数据库的理想选择，它提供一个可靠的数据管理和报告平台，其中包括安全的远程同步和管理功能
SQL Server Web（x86、x64）	对于为从小规模至大规模 Web 资产提供可扩展性和可管理性功能的 Web 宿主和网站来说，SQL Server Web 是一项拥有成本较低的选择

（续表）

术　语	定　义
SQL Server Express（x86 和 x64） SQL Server Express with Tools（x86 和 x64） SQL Server Express with Advanced Services（x86 和 x64）	SQL Server Express 数据库平台基于 SQL Server。它也可用于替换 Microsoft Desktop Engine（MSDE）。SQL Server Express 与 Visual Studio 集成，使开发人员可以轻松开发功能丰富、存储安全且部署快速的数据驱动应用程序 SQL Server Express 免费提供，且可以由 ISV 再次分发（视协议而定）。SQL Server Express 是学习和构建桌面及小型服务器应用程序的理想选择，也是独立软件供应商、非专业开发人员和热衷于构建客户端应用程序的人员的最佳选择。如果您需要使用更高级的数据库功能，则可以将 SQL Server Express 无缝升级到更复杂的 SQL Server 版本
Compact 3.5 SP1（x86）Compact 3.1（x86）	SQL Server Compact 3.5 免费提供，是生成用于基于各种 Windows 平台的移动设备、桌面和 Web 客户端的独立和偶尔连接应用程序的嵌入式数据库的理想选择

三、下载 SQL Server 2008

目前微软官网上只有最新版本 SQL Server 2014，用户购买其相应的软件后即可自行安装。下面给出一个 SQL Server 2008 服务器 Enterprise 版的安装过程，在购买的软件包中，找出安装文件“setup.exe”，如图 2-1 所示。

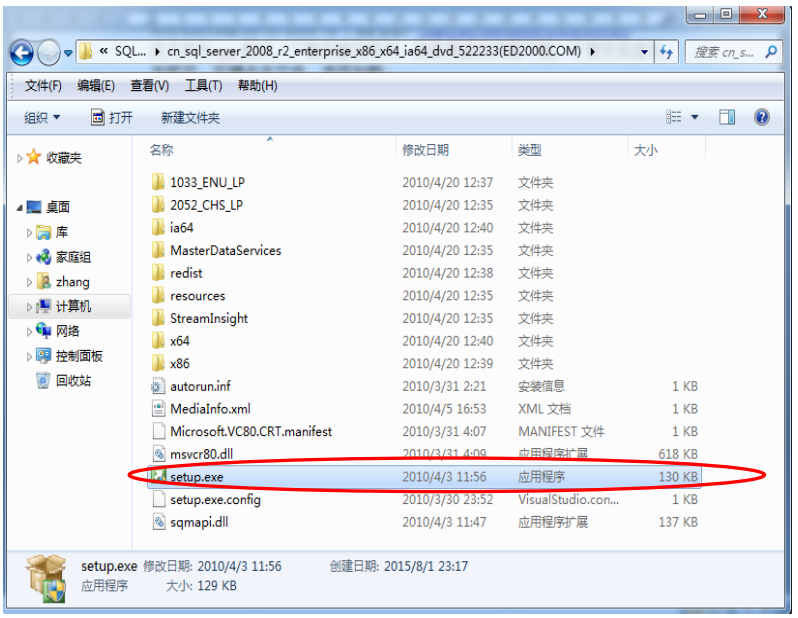


图 2-1　SQL Server 2008 安装文件信息

四、安装 SQL Server 2008

微软公司为 SQL Server 2008 的安装过程提供了一个友好的安装向导，利用该向导可以很方便地安装该软件。在如图 2-1 所示的窗口中双击“setup.exe”开始如图 2-2 安装程序窗口，在图 2-2 窗口中的左侧选中“安装”，然后在右侧的列表中单击“全新 SQL Server 独立安装或向现有安装添加功能”，继续进行系统检测，全部通过弹出如图 2-3 所示“安装程序支持规则”窗口。

图 2-2　安装程序窗口

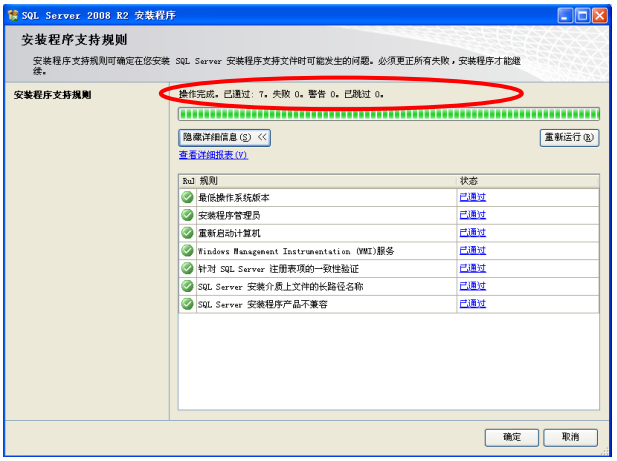


图 2-3　安装程序支持规则

在图 2-3 所示窗口中，如果系统检测状态均为“已通过”，则可单击“确定”按钮进入如图 2-4 所示的“产品密钥”窗口。

图 2-4　输入产品密钥窗口

在“产品密钥”窗口中，要求用户输入所购买该产品密钥，输入完成后，单击“下一步”按钮，则会弹出如图 2-5 所示的“许可条款”窗口。

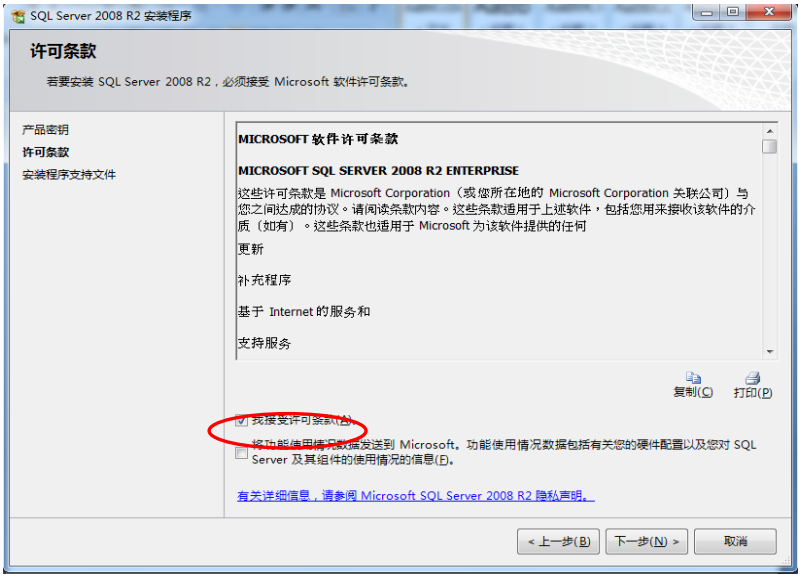


图 2-5　许可条款窗口

在“许可条款”窗口，选择“我接受许可条款”，然后单击“下一步”按钮，弹出如图 2-6 所示“安装程序支持规则”窗口。

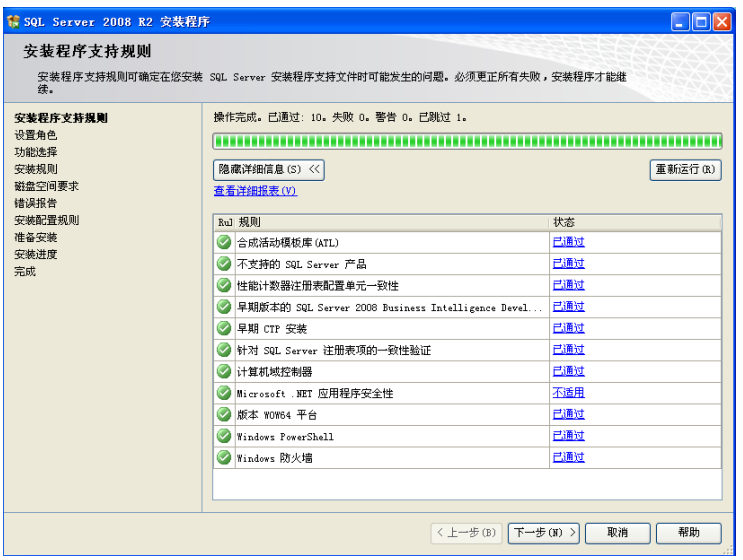


图 2-6　安装程序支持规则窗口

在“安装程序支持规则”窗口中，系统再次进行检测，全部通过后，单击“下一步”，弹出如图 2-7 所示的“设置角色”窗口。

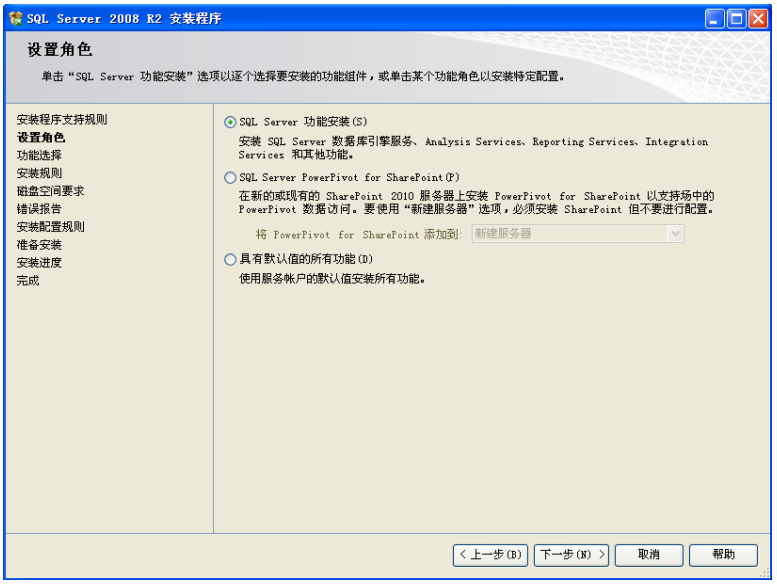


图 2-7　设置角色窗口

在“设置角色”窗口，直接单击“下一步”按钮，弹出如图 2-8 所示的“功能选择”窗口。

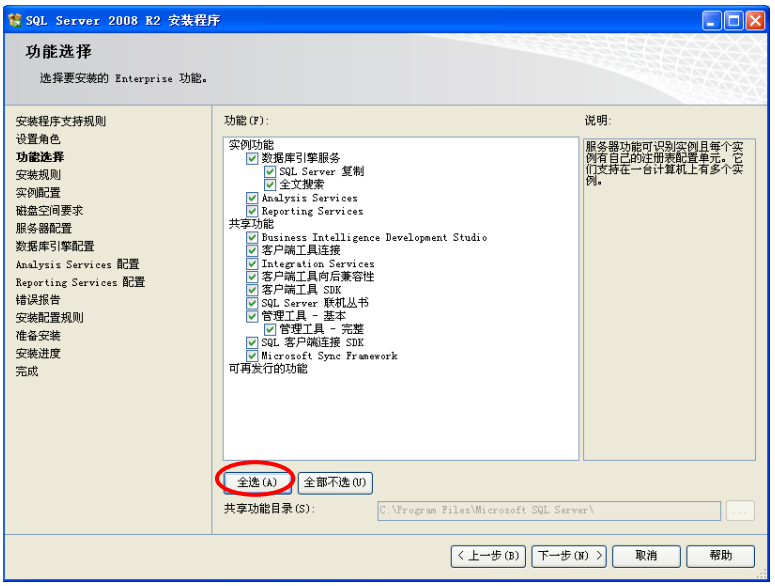


图 2-8　功能选择窗口

在“功能选择”窗口中，根据个人需要选择安装的功能，其中实例功能中的“数据库引擎服务”是 SQL Server 的核心服务，是必须要安装的。“Analysis Services”和“Reporting Services”分别是分析服务和报表服务，安装时可以根据需要选择。这里选择全选，安装位置根据个人机器情况进行必要的更改，然后单击“下一步”会弹出如图 2-9 所示的“安装规则”窗口。

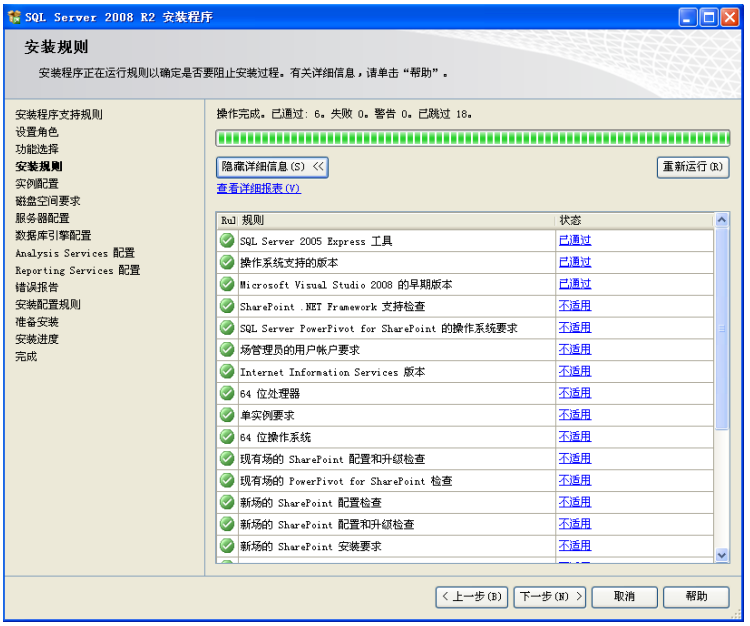


图 2-9　安装规则窗口

在“安装规则”窗口，安装程序运行规则以确定是否要阻止安装过程，运行结束后单击“下一步”按钮会弹出如图 2-10 所示的“实例配置”窗口。

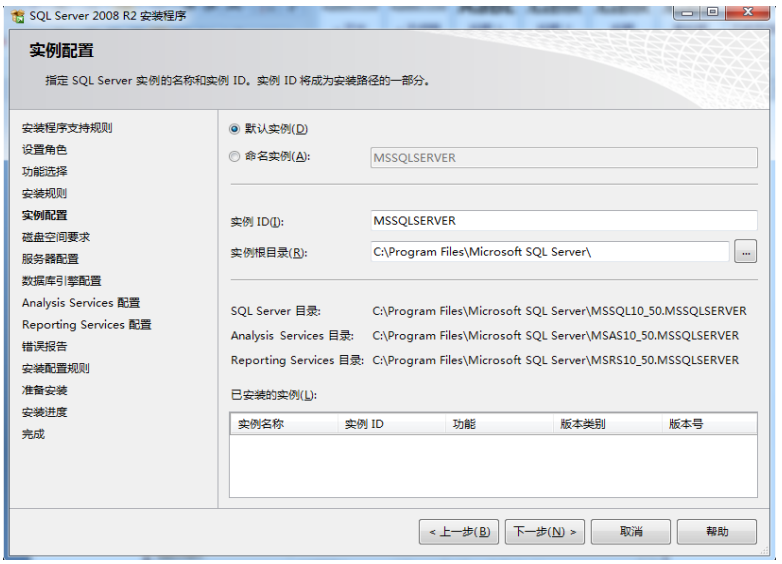


图 2-10　实例配置窗口

SQL Server 中所谓的实例指当在一台计算机上安装一次 SQL Server 时，就生成一个实例。默认实例的实例名就是安装 SQL Server 的计算机名，命名实例是指用户指定的一个实例名，这里选择默认实例。在“实例配置”窗口中，选择“默认实例”后“下一步”按钮，会弹出如图 2-11 所示的“磁盘空间要求”窗口。

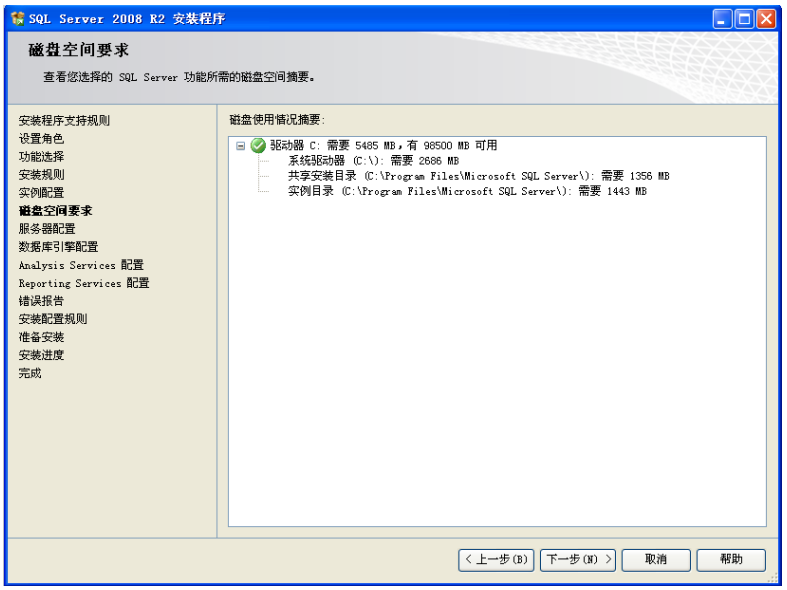


图 2-11　磁盘空间要求窗口

在“磁盘空间要求”窗口，直接单击“下一步”按钮进入如图 2-12 所示的“服务器配置”窗口。

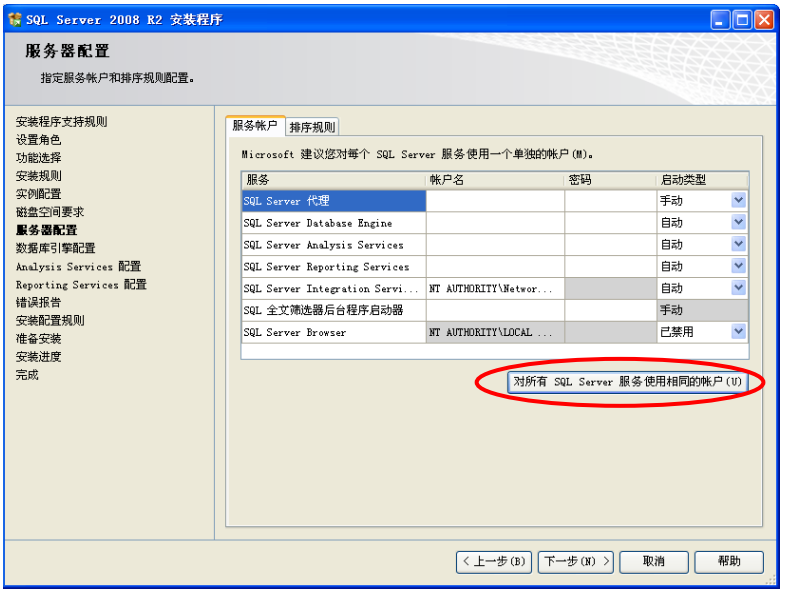


图 2-12　服务器配置窗口

在“服务器配置”窗口中的“服务账户”选项卡中可以为每个 SQL Server 服务配置账户、密码及启动类型。启动类型可以都设置成手动，需要时手工启动，如果设置成自动，则每次开机时都会启动相应的项目，会影响机器开机启动速度。在“服务器配置”窗口单击“对所有的 SQL Server 服务使用相同的账户”按钮，弹出对话框如图 2-13 所示。

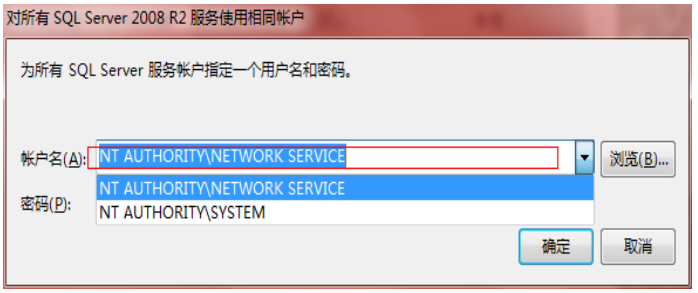


图 2-13　设置服务账户的用户名和密码对话框

在“设置用户名和密码”对话框中，“账户名”在下拉列表中进行选择“NT AUTHORITY\SYSTEM”，然后单击“确定”按钮，进入如图 2-14 所示的“数据库引擎配置”窗口。

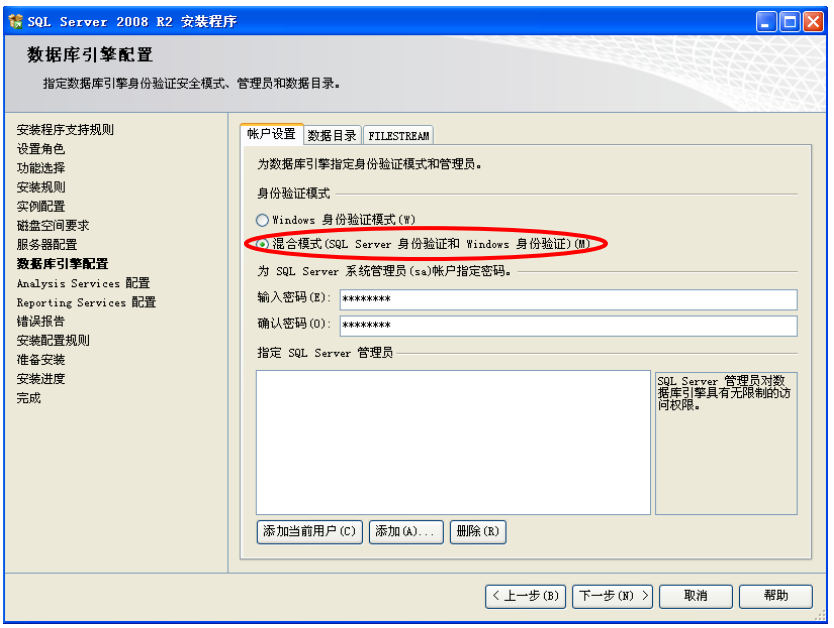


图 2-14　数据库引擎配置窗口

在“数据库引擎配置”窗口设置 SQL Server 服务器的认证模式，SQL Server 认证模式有“Windows 身份验证模式”和“混合模式”（SQL Server 身份验证和 Windows 身份验证）两种。

在 SQL Server 身份验证认证模式下，用户在连接 SQL Server 时必须提供 SQL Server 建立的合法登录名和登录密码，这些登录信息存储在系统表 syslogins 中，与 Windows 的登录账号无关。这里选择混合模式，SQL Server 认证模式在系统安装完成后也可以修改，在模块 4 的案例 3 相关知识部分将详细介绍。数据库引擎配置窗口还要设置数据库管理员的密码，同时，还要指定 SQL Server 的管理员，单击“添加当前用户”按钮，把当前用户设置为 SQL Server 的管理员。然后单击“下一步”按钮进入如图 2-15 所示的“Analysis Services 配置”窗口。

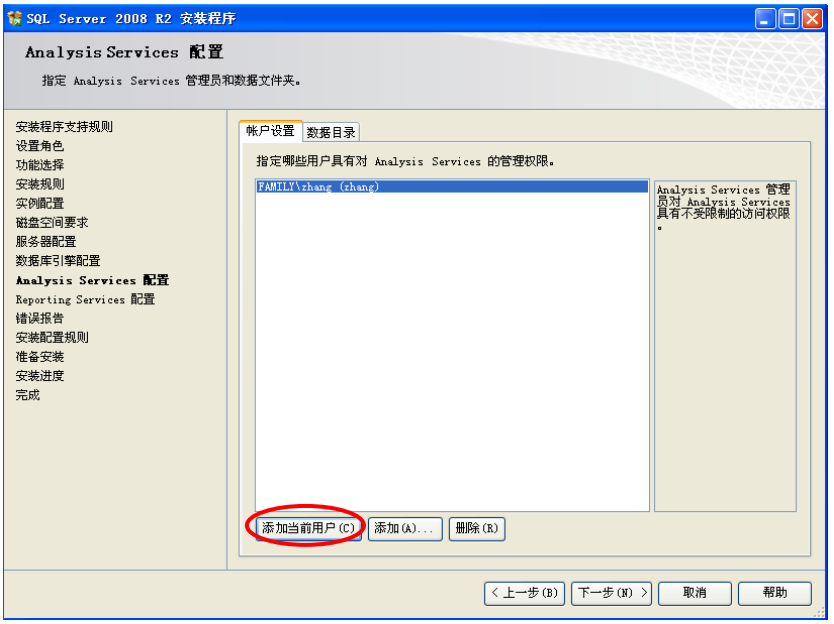


图 2-15　Analysis Services 配置窗口

在“Analysis Services 配置”窗口中，用户设置对 Analysis Services 具有管理权限的用户，单击“添加当前用户”，把当前用户设置为对 Analysis Services 具有管理权限的用户，然后单击“下一步”按钮，进入如图 2-16 所示的“错误报告”窗口。

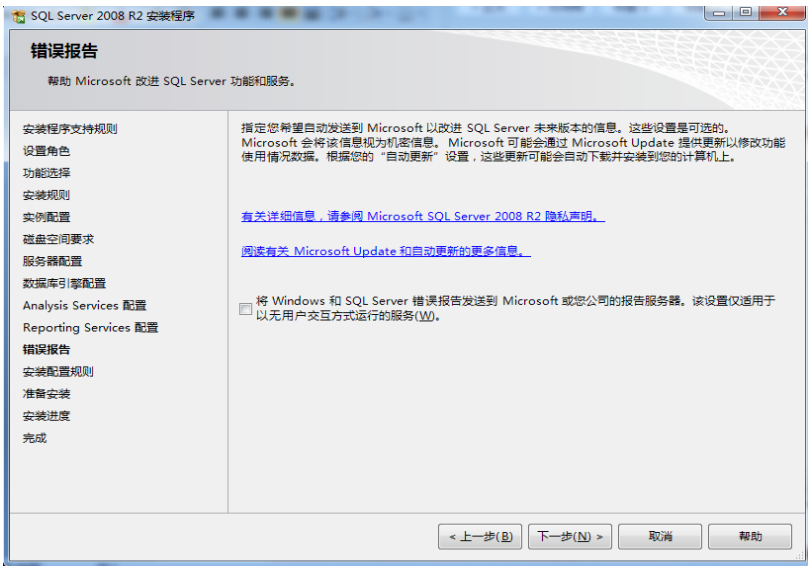


图 2-16　错误报告窗口

在“错误报告”窗口中，用户设置出现 Windows 或 SQL Server 错误信息时是否把错误信息发送到微软公司或公司的错误报告中。此项可以忽略，直接单击“下一步”按钮，进入如图 2-17 所示的“安装配置规则”窗口。

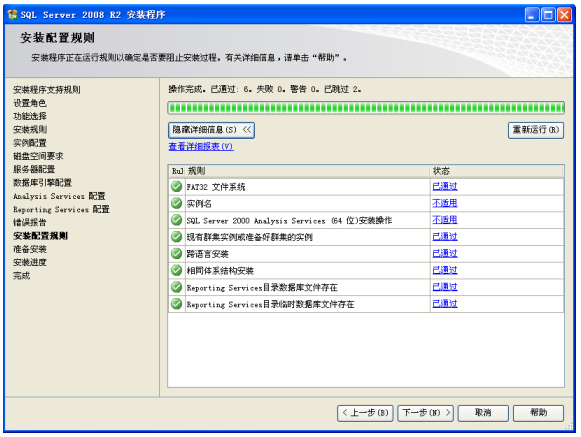


图 2-17　安装配置规则窗口

在“安装配置规则”窗口显示安装配置规则的信息，直接单击“下一步”按钮，进入如图 2-18 所示的“准备安装”窗口。

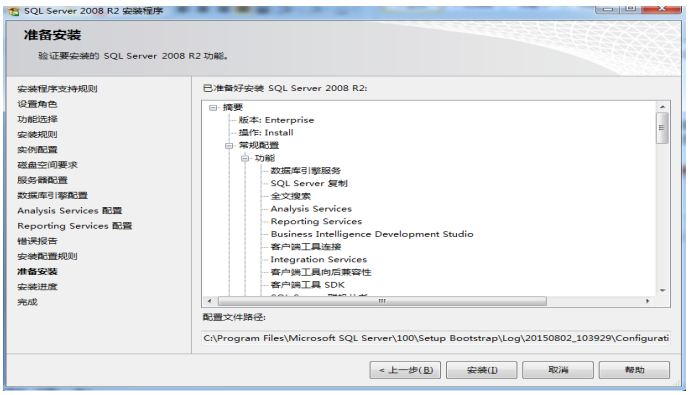


图 2-18　准备安装窗口

在“准备安装”窗口，系统列出了用户所选择的功能、实例配置、安装位置等信息。前面几步是一些配置信息和用户信息的设置，现在马上开始正式安装，单击“安装”按钮开始安装，整个安装过程大概需要 40 分钟到 1 个小时之间。安装完成后会弹出一个如图 2-19 所示的安装完成窗口。

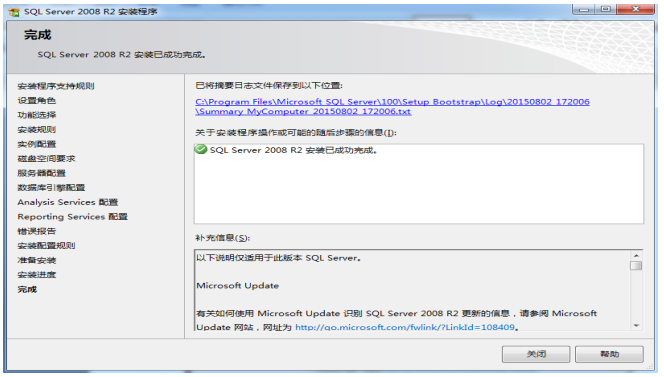


图 2-19　完成安装窗口

当出现“安装完成”窗口，提示用户 SQL Server 2008 的安装工作已经完成。打开控制面板中的服务如图 2-20 所示，可以看到有关 SQL Server 的相关服务，其中 SQL SERVER（MSSQLSERVER）需要启动，其他项目用到时再启动，以免影响机器速度。所有项目启动方式最好都设置为手动启动，用到时手动启动相关服务，如果设置为自动启动会影响开机速度。

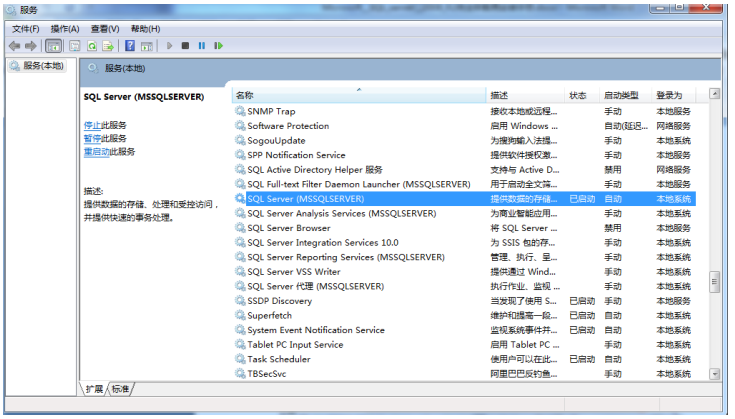


图 2-20　服务中有关 SQL Server 2008 相关项

五、配置 SQL Server 2008

成功安装好 SQL Server 2008 后，根据需要对 SQL Server 2008 的服务器端和客户端进行适当的配置，以更符合用户的要求，下面将介绍使用配置管理器工具配置 SQL Server 2008 的方法。安装完 SQL Server 2008 后，可以通过单击“开始”→“所有程序”→“Microsoft SQL Server 2008 R2”→“配置工具”→“SQL Server 配置管理器”，打开“SQL Server 2008 配置管理器”工具如图 2-21 所示。配置管理器的管理项目分为三部分：

（1） SQL Server 服务管理。

（2） SQL Server 网络配置。

（3） SQL 客户端配置。

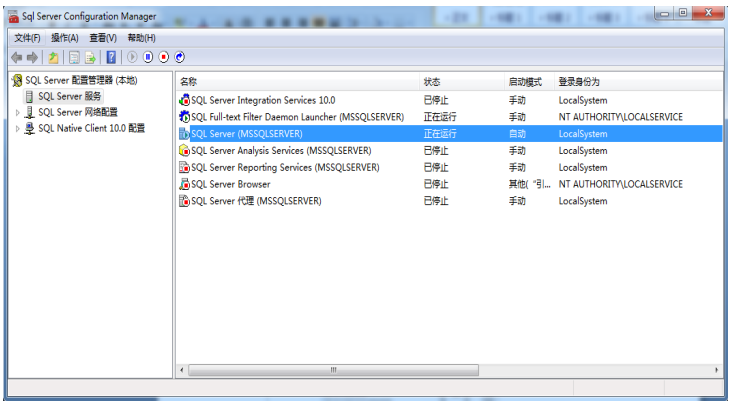


图 2-21　SQL Server 2008 配置管理器

1. 服务管理

SQL Server 是一个大型数据库产品，在服务器后台要运行许多不同的服务，完整安装的 SQL Server 包括 9 个服务，如图 2-22 所示，这 9 个服务在控制面板的服务中可以看到，其中 7 个服务可使用 SQL Server 配置管理器来管理，另两个作为后台支持的服务。

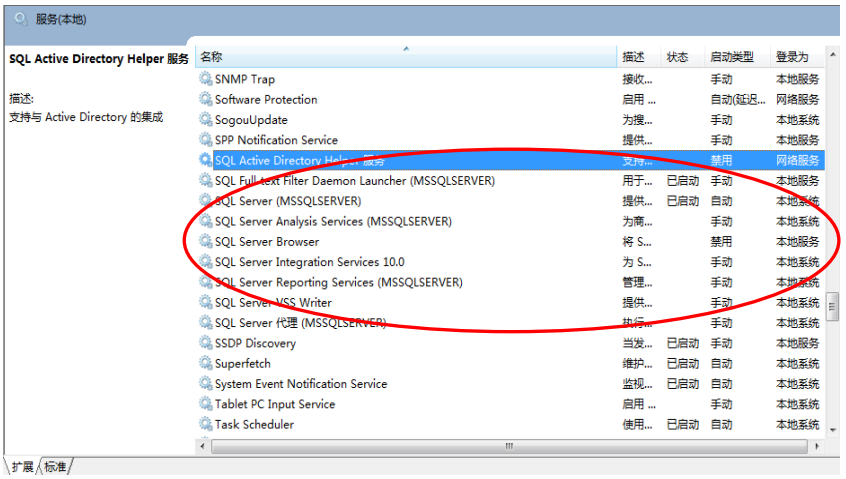


图 2-22　服务中 SQL Server 2008 中 9 个服务项目

SQL Server 配置管理器管理的 7 个服务包括：

（1） 集成服务（SQL Server Integration Services）：支持 Integration Services 引擎。

（2） 全文（SQL Full- Text Filter Daemon Launcher）目录：支持文本搜索功能。

（3） SQL Server：核心数据库引擎，其功能包括 SQL Server 数据存储、查询和系统配置。

（4） 分析服务（SQL Server Analysis Services）：支持 Analysis Services 引擎。

（5） 报表服务（SQL Server Reporting Services）：支持 Reporting Services 的底层引擎。

（6） SQL Server Browser：支持通告服务器，通过浏览局域网可以确认系统是否安装了 SQL Server。

（7） SQL Server 代理（SQL Server Agent）：SQL Server 中作业调度的主引擎。利用该服务，可以按照不同调度安排作业。这些作业可有多个任务，甚至可以根据先前任务的结果来分解成不同子任务。SQL Server Agent 运行的示例包括备份以及例程输入与输出任务。

上面的 7 个服务项目不都需要启动，其中的核心数据库引擎是核心服务，只有启动了该服务，SQL Server 数据库管理系统才能发挥其作用，用户也才能建立与 SQL Server 数据库服务器的连接，其他服务项目需要时再启动。

通常，通过 SQL Server 配置管理器来启动或停止相关服务项目，或更改其启动模式，具体操作时：在对应服务项目上右击，选择属性，或直接双击该服务项目。比如在 SQL Server（MS SQLSERVER）服务上右击，会弹出如图 2-23 所示的“属性设置”窗口。

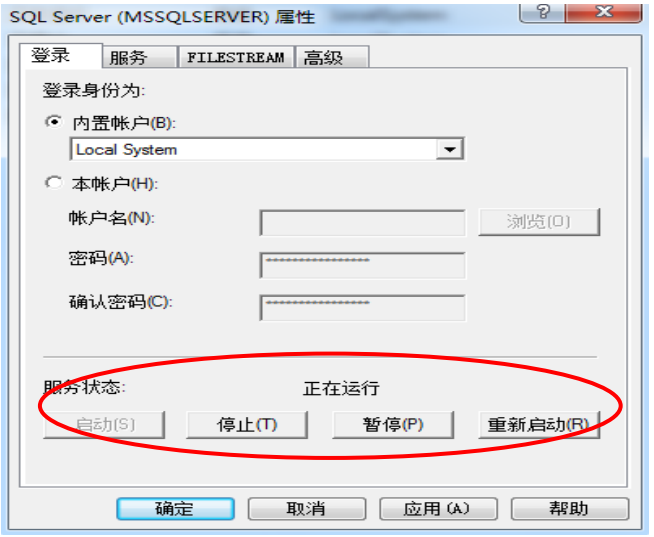


图 2-23　属性设置窗口

在“属性设置”窗口的“登录”选项卡中，若服务已经“启动”，则可以对此项服务选择“停止”、“暂停”或“重新启动”。

在“属性设置”窗口的“服务”选项卡中，如图 2-24 所示“服务”选项卡中可以设置其启动模式为“自动”、“手动”或“已禁用”。其中：

（1） 自动：表示每当操作系统启动时自动启动该服务。

（2） 手动：表示需要用户手工启动该服务。

（3） 已禁用：表示禁止该服务启动。

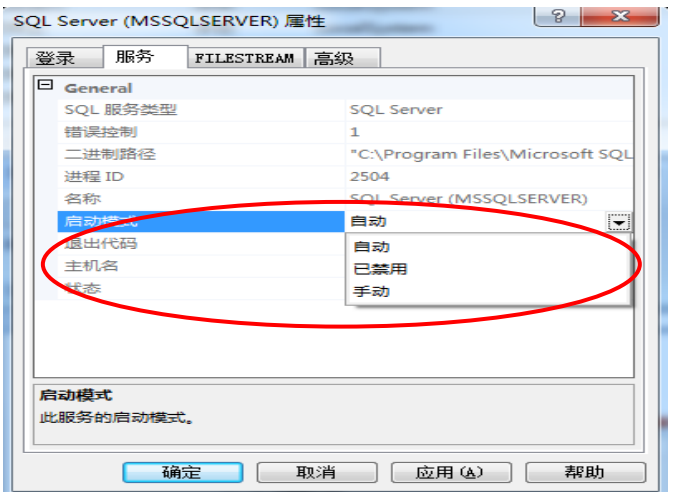


图 2-24　服务选项卡对应窗口

2. 网络配置

在图 2-21 所示的“SQL Server Configuration Manager”窗口中，展开“SQL Server 网络配置”节点左边的加号，然后单击其下的“MSSQLSERVER 的协议”，在窗口右侧将显

示 SQL Server 提供的网络协议，如图 2-25 所示的“SQL Server 2008 网络配置”窗口。

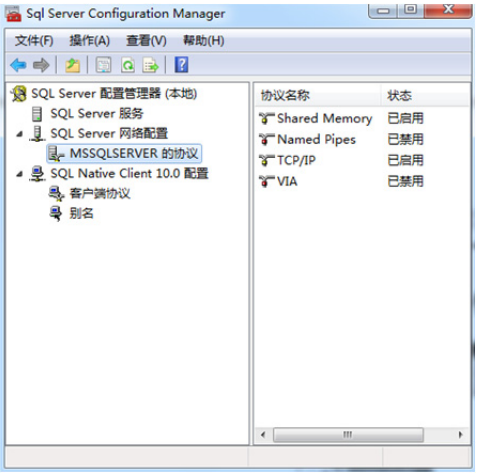


图 2-25　SQL Server 2008 网络配置窗口

SQL Server 2008 提供了 Shared Memory（共享内存），Named Pipes（命名管道）、TCP/IP 和 VIA 四种网络协议，这 4 种网络协议的基本含义如下：

（1） Shared Memory：它是可供使用的最简单的协议，该协议只在本地访问服务器时有用（即 Web 服务器与数据库安装在相同的服务器上），可以大大提高系统性能。

（2） Named Pipes：在 TCP/IP 不可用或者没有域名服务（Domain Name Service DNS）服务器用于对 TCP/IP 下的服务器命名时，命名管道非常有用。某个进程使用一部分内存来向另一个进程传递信息，因此一个进程的输出就是另一个进程的输入。

（3） TCP/IP：它是互联网上使用最广泛的通用协议，已经成为事实上的标准网络协议，如果想要通过 Internet 直接连接到 SQL Server，则 TCP/IP 协议是唯一的选项。通过该协议可以与互联网中硬件结构和操作系统不同计算机进行通信。其中包括路由网络流量的标准，并能够提供高级安全功能。

（4） VIA：虚拟接口适配器（VIA）协议与 VIA 硬件一同使用。VIA 协议是 SQL Server 2005 新推出的协议，适合在局域网内部使用。

3. 客户端配置

在图 2-21 所示的“SQL Server Configuration Manager”窗口中，展开“SQL Native Client10.0 配置”节点，然后单击其中的“客户端协议”节点，将出现如图 2-26 所示的“配置客户端协议”窗口。

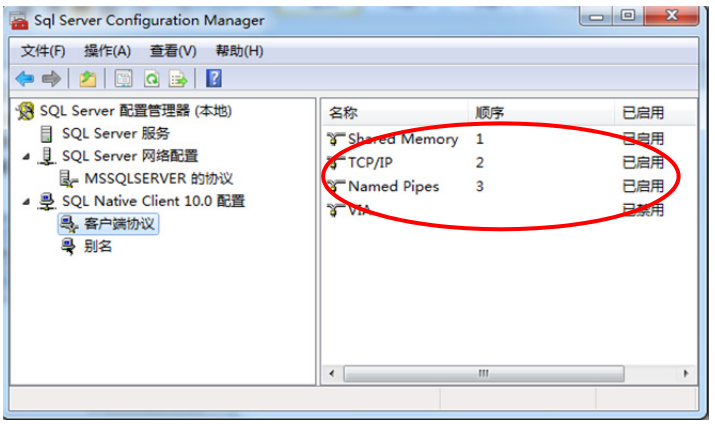


图 2-26　配置客户端协议窗口

在图 2-26“配置客户端协议”窗口中，可以看到当前客户端已启用了“Shared Memory”、“TCP/IP”和“Name Pipes”三种协议，也就是说，如果服务器端的网络配置中，启用了上述 3 中协议中的任何一种，那么客户端就可以连接到服务器上。用户可以在某个协议上右击，然后在弹出的快捷菜单中选择“启用”或“禁用”命令以启用或禁用某个协议；SQL Server 提供了一项功能，即客户机可以先使用一个协议，如果该协议不起作用，则再使用另一协议。在图 2-26“配置客户端协议”窗口中右击客户端协议，会弹出如图 2-27 所示的“客户端协议属性”窗口。

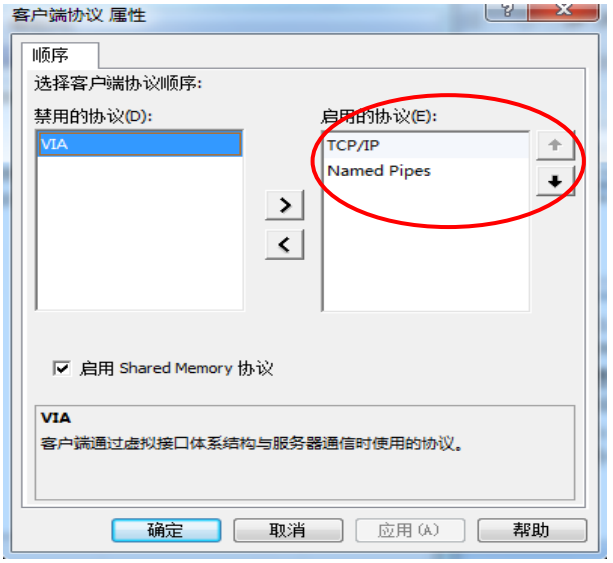


图 2-27　客户端协议属性窗口

在“客户端协议 属性”窗口的“顺序”选项卡中可以通过上下箭头更改协议的启动顺序。默认情况下，SQL Server 使用“共享内存”协议连接到 SQL Server 的本地实例，使用“TCP/IP”或“命名管道”连接到其他计算机上的 SQL Server 实例。

特别说明：如果网络支持 TCP/IP 协议，则对于任何远程访问，配置服务器使用它，IP 协议的系统开销更少，而运行速度更快；但该协议对本地服务器没有意义，如果不需要通过网络来查看本地 SQL Server，则使用 Shared Memory NetLib 速度更快。

别名是用于引用此连接的名称，是可用于进行连接的备用名称。别名封装了连接字符串所必需的元素，并使用用户所选择的名称显示这些元素。请在以下情况下创建别名：使用 TCP/IP、命名管道或 VIA 并且希望提供自定义连接字符串时；希望使用服务器名称之外的其他名称进行连接时。

相关知识

一、卸载 SQL Server 2008

如果要卸载 SQL Server 2008（如果已经启动服务），则首先停止 SQL Server 2008 的全部服务，然后再执行卸载操作。

停止 SQL Server 2008 服务的方法如下：

在 SQL Server 2008 配置管理器中或控制面板的服务中，需要停止的 SQL Server 服务上右击，在弹出的快捷菜单中选择“停止”命令，即可停止选中的服务。

SQL Server 2008 本身没有提供卸载工具，需要借助 Windows 操作系统的控制面板来实现，或通过软件进行卸载，下面给出 Win7 环境下通过控制面板卸载 SQL Server 2008 的过程：

（1） 打开 Windows 操作系统的“控制面板”，单击“程序和功能”图标，打开程序和功能窗口。

（2） 在“程序和功能”列表框中选择“Microsoft SQL Server 2008 R2”，如图 2-28 所示，右击，弹出“卸载/更改”选项，单击“卸载/更改”选项，会弹出图 2-29“添加修复删除选项”窗口。

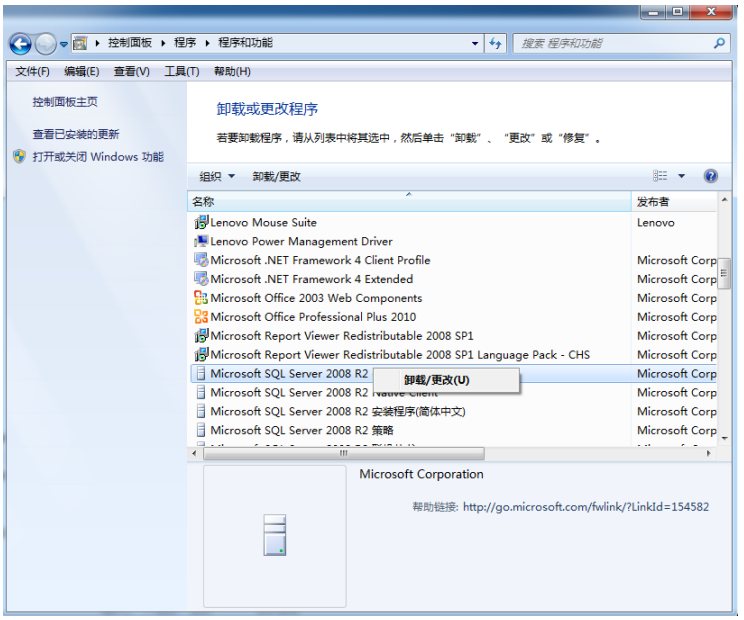


图 2-28　程序和功能窗口

图 2-29　添加修复删除选项对话框

在“添加修复删除选项”对话框，选择“删除”后弹出如图 2-30 所示“安装程序支持规则”窗口。

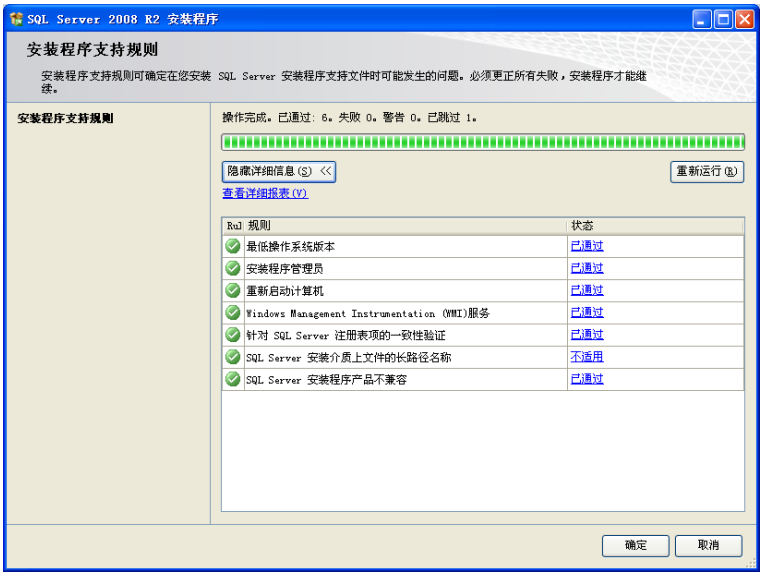


图 2-30　安装程序支持规则窗口

（3） 开始卸载过程，在安装支持规则窗口中，全部通过或者跳过即没问题，单击“确定”按钮，如果“重新启动计算机”不通过，则重启计算机，再进行卸载即可。

（4） 在安装支持规则窗口中单击“确定”按钮后，弹出如图 2-31 所示“选择实例”窗口。

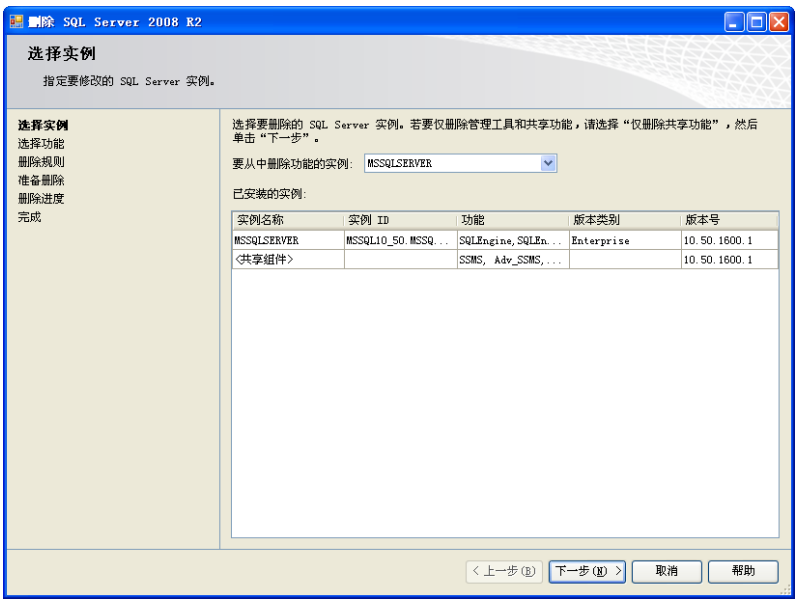


图 2-31　选择实例窗口

在“选择实例”窗口中，选择要删除的实例，然后单击“下一步”按钮，会弹出如图 2-32 所示“选择功能”窗口。

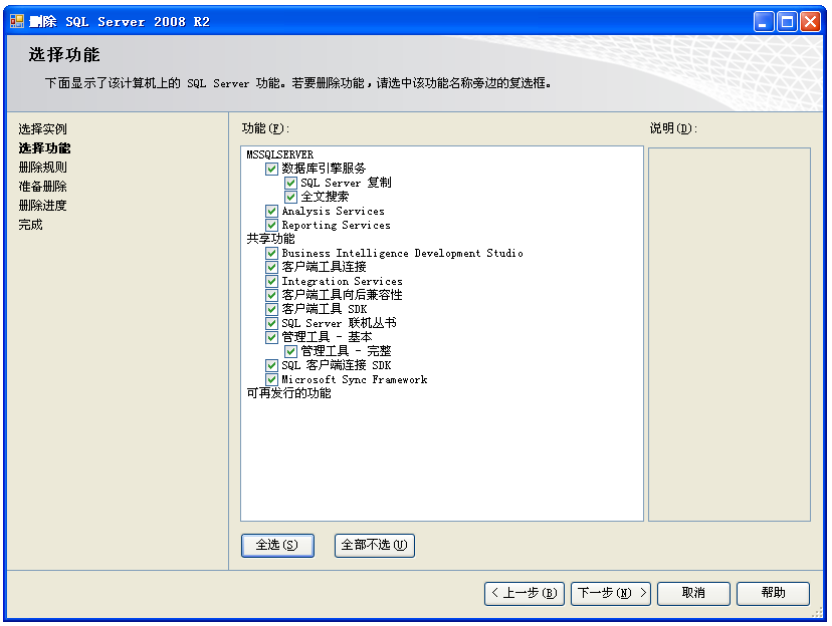


图 2-32　选择功能对话框

（5） 在选择功能窗口中选择“全选”按钮，删除所有已经安装的 SQL Server 2008 选项。然后单击“下一步”按钮弹出如图 2-33 所示的“删除规则”窗口。

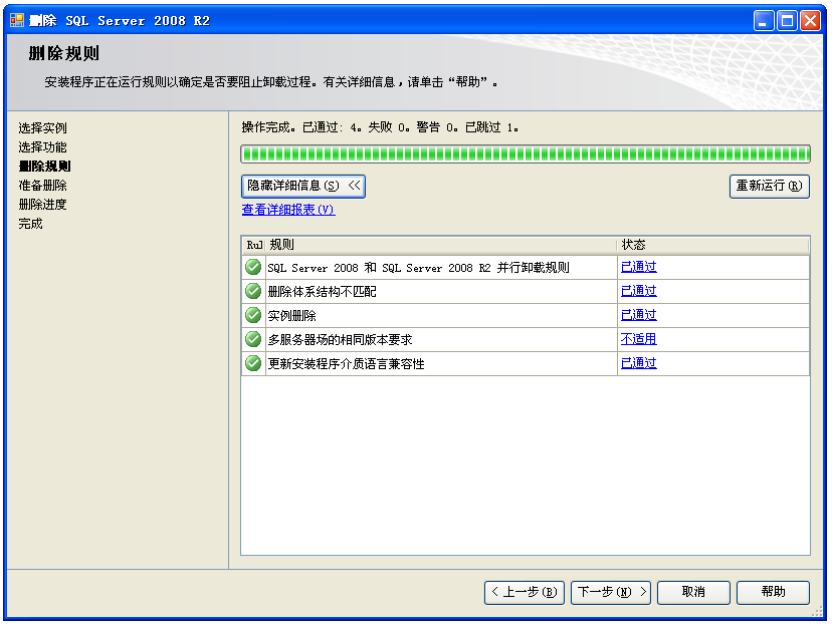


图 2-33　删除规则对话框

（6） 在“删除规则”窗口中选择“下一步”按钮弹出如图 2-34 所示的“准备删除”窗口，单击“删除”按钮开始删除。

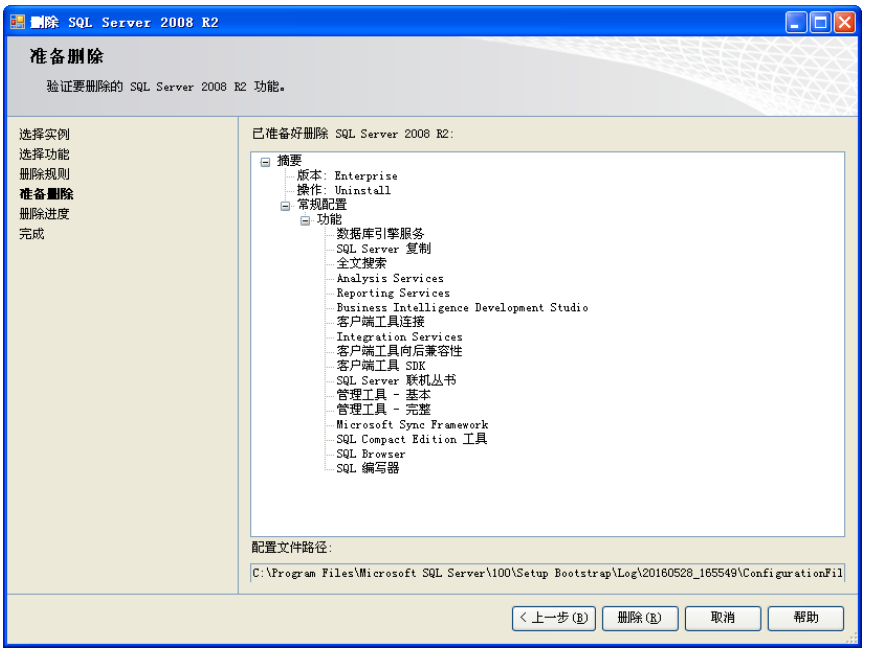


图 2-34　准备删除对话框

（7） 删除完成，等待删除完成后会弹出如图 2-35 所示的删除“完成”窗口。

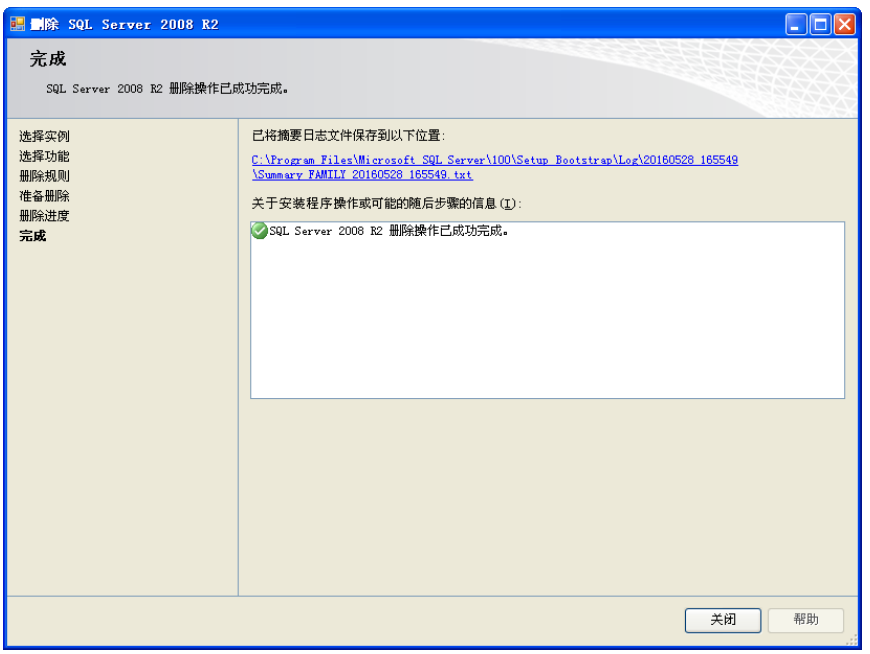


图 2-35　删除完成对话框

（8） 然后进入控制面板，右击删除里面所有有关 Microsoft SQL server 的项目，如图 2-36 所示。

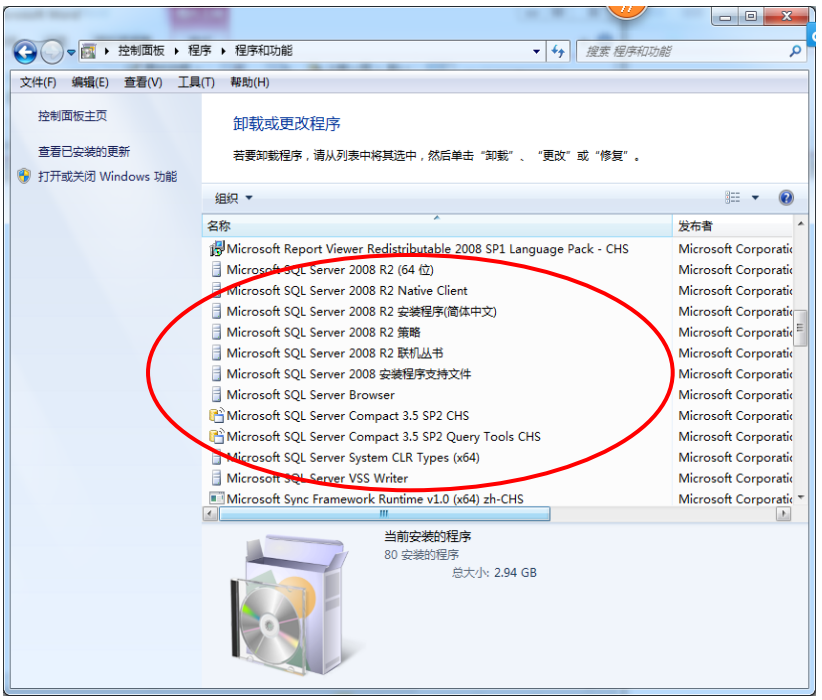

控制面板 ▸ 程序 ▸ 程序和功能
搜索 程序和功能
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
控制面板主页
查看已安装的更新
打开或关闭 Windows 功能
卸载或更改程序
若要卸载程序，请从列表中将其选中，然后单击“卸载”、“更改”或“修复”。
组织
名称
发布者
Microsoft Report Viewer Redistributable 2008 SP1 Language Pack - CHS
Microsoft SQL Server 2008 R2 (64 位)
Microsoft SQL Server 2008 R2 Native Client
Microsoft SQL Server 2008 R2 安装程序(简体中文)
Microsoft SQL Server 2008 R2 策略
Microsoft SQL Server 2008 R2 联机丛书
Microsoft SQL Server 2008 安装程序支持文件
Microsoft SQL Server Browser
Microsoft SQL Server Compact 3.5 SP2 CHS
Microsoft SQL Server Compact 3.5 SP2 Query Tools CHS
Microsoft SQL Server System CLR Types (x64)
Microsoft SQL Server VSS Writer
Microsoft Sync Framework Runtime v1.0 (x64) zh-CHS
Microsoft Corporati
当前安装的程序
80 安装的程序
总大小: 2.94 GB

图 2-36　控制面板中 Microsoft SQL server 的项目

（9） 删除安装盘中残余文件，本次安装默认在 C 盘，路径如下所示。

C:\Program Files\Microsoft SQL Server

C:\ 用户\Administrator\AppData\Local\Microsoft

C:\ 用户\Administrator\AppData\Roaming\Microsoft

说明：Windows 7 的 C 盘下“用户”文件夹在 Windows 8 显示为 Users，Administrator 即计算机管理员账户，具体根据计算机自定义的名称。如果再删除文件夹的时候，发现删除不掉，提示被占用，可以选择重启计算机试试，如果还是不行，可以采取其他强制的方法。

（10） 删除注册表残余项目，运行注册表，删除如下项。

HKEY_CURRENT_USER\Software\Microsoft\Microsoft SQL Server

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer

路径具体操作是单击“开始”，然后在“搜索程序和文件”中输入“regedit”，如图 2-37 所示，回车后会弹出如图 2-38 所示的注册表编辑器窗口。展开相应文件夹找到对应项进行删除。

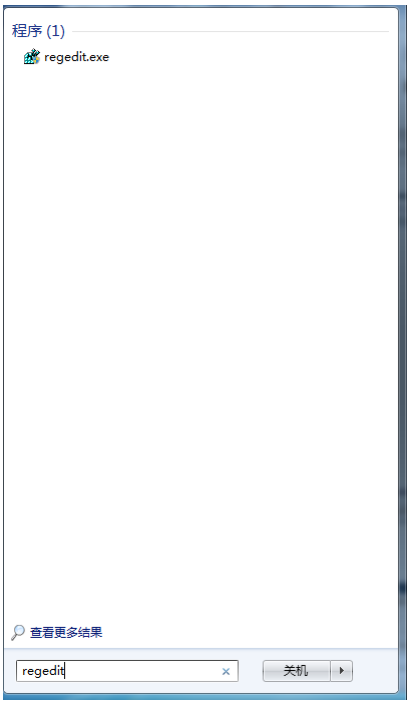


图 2-37　搜索中输入 regedit 对话框

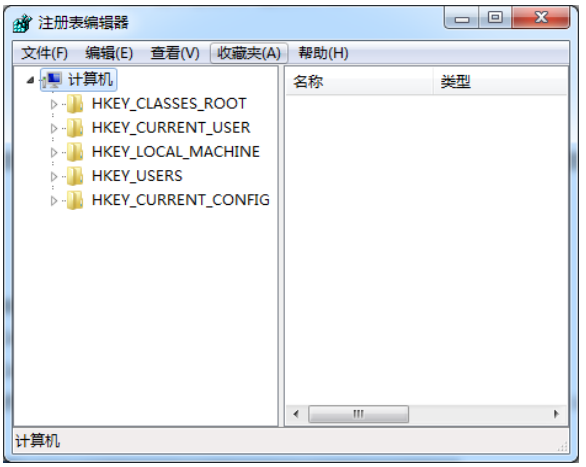


图 2-38　注册表编辑器

二、SQL Server 2008 工具简介

1. SQL Server Management Studio 工具简介

SQL Server Management Studio 是 SQL Server 2008 中最重要的管理工具之一，使用这个工具既可以用图形化的方法，也可以通过编写 SQL 语句来实现对数据库的操作。SQL Server Management Studio 是一个集成环境，用于访问、配置、管理和开发 SQL Server 的所有组件。SQL Server Management Studio 组合了大量图形工具和丰富的脚本编辑器，使各种技术水平的开发人员和管理员都能访问 SQL Server。

1） 启动 SQL Server Management Studio。

“开始”→“所有程序”→“Mircrosoft SQL Server 2008 R2”→“SQL Server Management Studio”，打开 SQL Server Management Studio（SSMS），弹出如图 2-39 的“连接到服务器”窗口。

在“连接到服务器”窗口中，各选项含义如下：

（1） 服务器类型：列出了 SQL Server 2008 数据库服务器所包含的服务，其下拉列表框中列出的内容就是安装时选择安装的服务，用户可以根据使用需要，连接到不同的服务。这里选择“数据库引擎”，即 SQL Server 服务。

（2） 服务器名称：指定要连接的数据库服务器的实例名。SSMS 能够自动扫描当前网络中的 SQL Server 实例。如果“服务器名称”中列出的不是希望连接的 SQL Server 实例，则可单击右侧下拉式列表按钮，然后在列表框中选择“浏览更多”命令，弹出如图 2-40 所

示的“查找服务器”窗口。

图 2-39　连接到服务器对话框

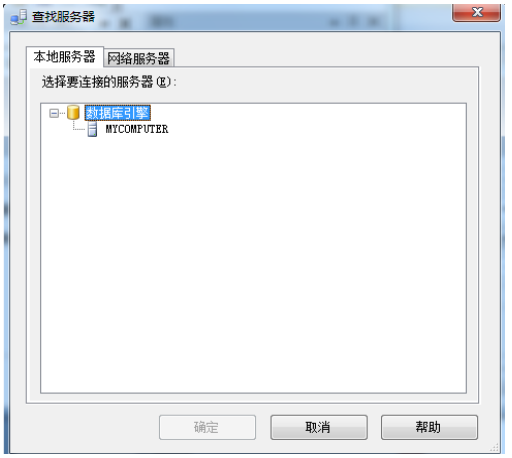


图 2-40　查找服务器对话框

在此窗口中选中要连接的服务器实例，然后单击“确定”按钮。也可以在“服务器名称”文本框中手动输入要连接的服务器实例名。

（3） 身份验证：选择使用哪种身份的用户连接到数据库服务器实例，这里有两种选择“Windows 身份验证”和“SQL Server 身份验证”。如果选择的是“Windows 身份验证”则表示当前登录到 Windws 的用户连接到数据库服务器实例；若选择的是“SQL Server 身份验证”，则需要输入 SQL Server 合法的登录名和相应密码。

2） SQL Server Management Studio 工具的使用。

连接成功后，进入 SSMS 操作窗口，如图 2-41 所示。

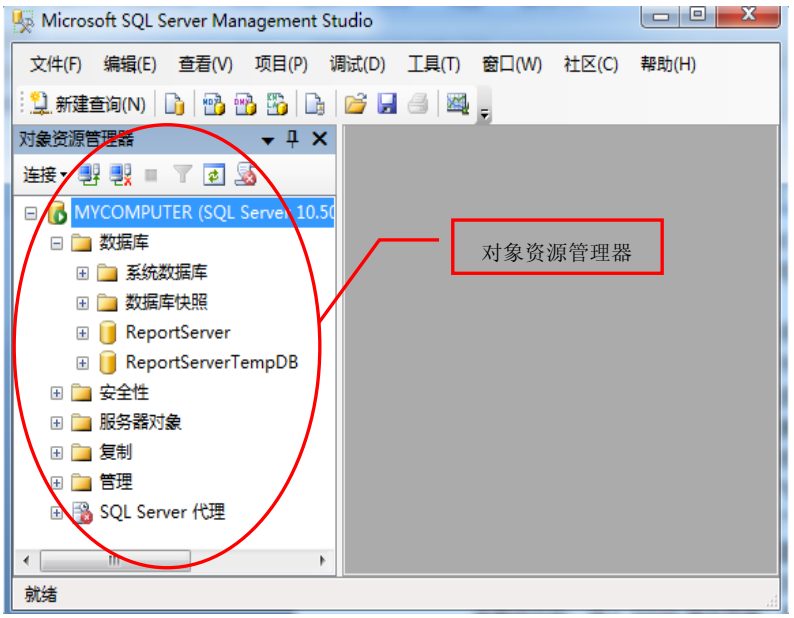


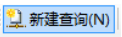
图 2-41　SSMS 操作窗口

SSMS 操作窗口主要由两部分组成，对象资源管理器和查询编辑器，这两部分是把 SQL Server 2000 的 Enterprise Manager（企业管理器）和 Query Analyzer（查询分析器）两个工具结合在一个窗口上，这样可以在对服务器进行图形化管理的同时编写 Transact SQL 脚本，且用户可以直接通过 SQL Server 2008 的“对象资源管理器”窗口来操作数据库，下面分别给予介绍。

（1） 对象资源管理器。

对象资源管理器默认在 SSMS 窗口的左侧，如图 2-41 所示，是操作数据库的主要工具，利用对象资源管理器可以查看数据库对象，以 Windows 身份验证模式登录到 SQL Server Management Studio。在对象资源管理器中展开“数据库”，会看到当前连接的服务器中所有的数据库和系统数据库，选择系统数据库中的 master 数据库并展开，则将列出该数据库中所包含的所有对象，如表、视图、同义词、存储过程等，这些对象我们后面都要一一介绍。我们查看已经建立的数据库和数据库的对象就是通过这个窗口来实现的。

（2） 查询编辑器。

在图 2-41 中单击“新建查询”  图标按钮，可以打开查询编辑器窗口，如图 2-42 所示。

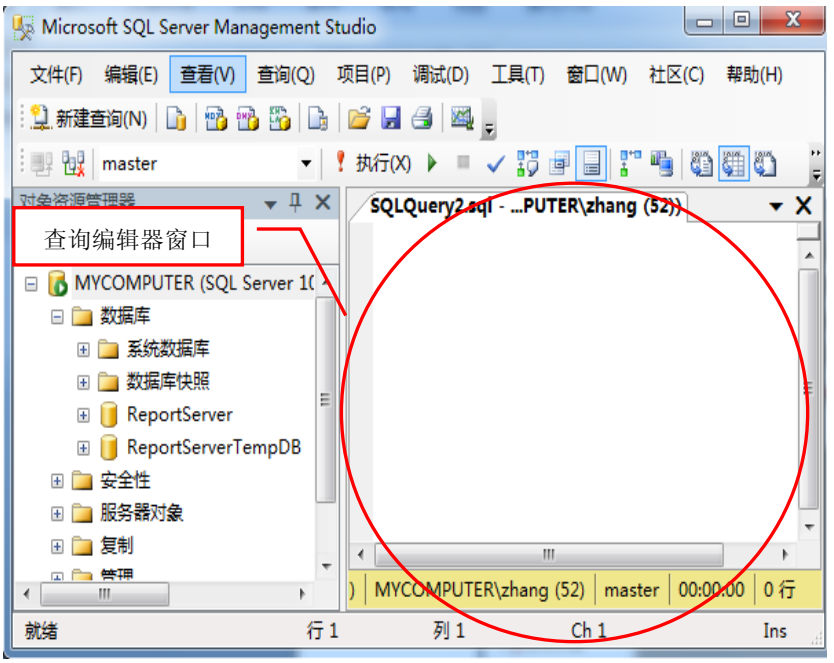


图 2-42 查询编辑器窗口

单击“新建查询”按钮右侧的“数据库引擎查询”图标按钮，将打开图 2-39 所示的窗口，在此窗口中，用户可以指定在查询编辑器窗格中执行操作的用户。

单击“对象资源管理器”中的“连接” 图标按钮。在出现的下拉列表框中选择“数据库引擎”命令如图 2-43 所示，也将弹出图 2-39 窗口，利用此窗口，用户可以再建立一个到数据库服务器的连接。

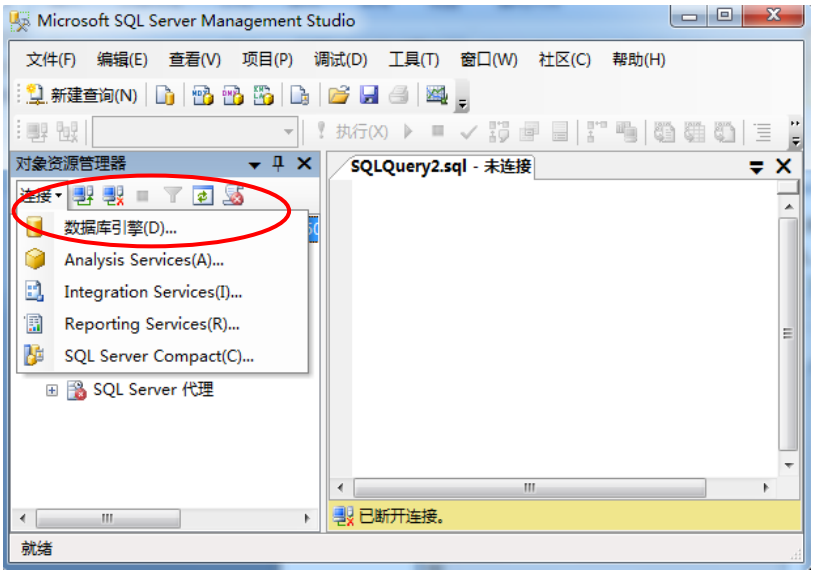


图 2-43　连接数据库引擎窗口

SSMS 工具提供了图形化的操作窗口来创建和维护对象，同时也提供了用户编写 T_SQL 语句、并通过执行 SQL 语句创建和管理对象的工具即查询编辑器。查询编辑器以选项卡窗口的形式存在于 SSMS 窗口右侧的文档窗格中，可以通过如下方式之一打开查询编辑器。

① 单击标准工具栏上的图标按钮 新建查询(N)。

② 单击标准工具栏上的“数据库引擎查询”图标按钮。

③ 选择“文件”菜单中“新建”命令下的“数据库引擎查询”命令。

查询编辑器打开后会出现一个编辑窗口如图 2-42 所示，在该编辑窗口可以输入 SQL 指令。查询编辑器工具栏如图 2-44 所示。

图 2-44　查询编辑器工具栏

在图 2-44 中，最左边的两个图标按钮用于处理到服务器的连接。其中，第一个图标按钮是“连接”图标按钮，用于请求一个到服务器的连接，如果当前没有建立任何连接的话，则单击此图标按钮将弹出如图 2-39 所示的窗口。如果当前已经建立到服务器的连接，则此图标按钮为不可用状态。第二个图标按钮是“更改连接”图标按钮，单击此图标按钮表示要更改当前的连接。“更改连接”图标按钮的右侧是一个下拉列表框 master，该列表框列出了当前查询编辑器所连接的数据库服务器上的所有数据库，列表框上显示的数据库是当前连接正在操作的数据库。如果要在不同的数据库上执行操作，可以在列表框中选择不同的数据库，选择一个数据库就代表要执行的 SQL 指令都是在此数据库上进行的。

随后的 4 个图标按钮与查询编辑器中所键入的代码的执行有关。“执行”图标按钮

执行(X)用于执行在编辑区所选中的代码，如果没有选中任何代码，则表示执行编辑器中全部代码。“调试”图标按钮用于对代码进行调试。“取消执行查询”按钮用户取消正在执行的查询，如果没有查询执行，则该按钮显示不可用状态，在执行代码时它将成为红色。“分析”图标按钮用于对编辑区中选中的代码，如果没有选中任何代码，则表示分析编辑器中全部代码，进行语法分析。

图标按钮组用于改变查询结果的显示形式。这组按钮从左到右分别是：设置查询结果按文本格式显示、设置查询结果按网格形式显示和设置查询结果保存到一个文件中。图标按钮用于为选中的代码行添加注释，图标按钮用于取消对选中行代码的注释。图标按钮用于减少选中代码的缩进，图标按钮用于增加选中代码的缩进。

2. 模板资源管理器和解决方案管理器

SSMS 除了两个重要对象外，还有模板资源管理器和解决方案管理器，下面分别给予简单介绍。

（1） 模板资源管理器。

在 SQL Server Management Studio 的查询分析器中使用 Transact SQL 脚本可以查询到对象建立的所有任务。而使用脚本编制数据库对象与图形化向导编制数据库对象相比，最大的优点是：使用脚本化方式具有图形化向导方式所无法比拟的灵活性。但是，高度的灵活性也就意味着使用它的时候有着比图形化向导方式更高的难度。SQL Server Management Studio 提供了模板资源管理器来降低编写脚本的难度。在 SQL Server Management Studio 的菜单栏中单击“查看”，选择“模板资源管理器”，窗口右侧将出现“模板资源管理器”窗格，如图 2-45 所示。

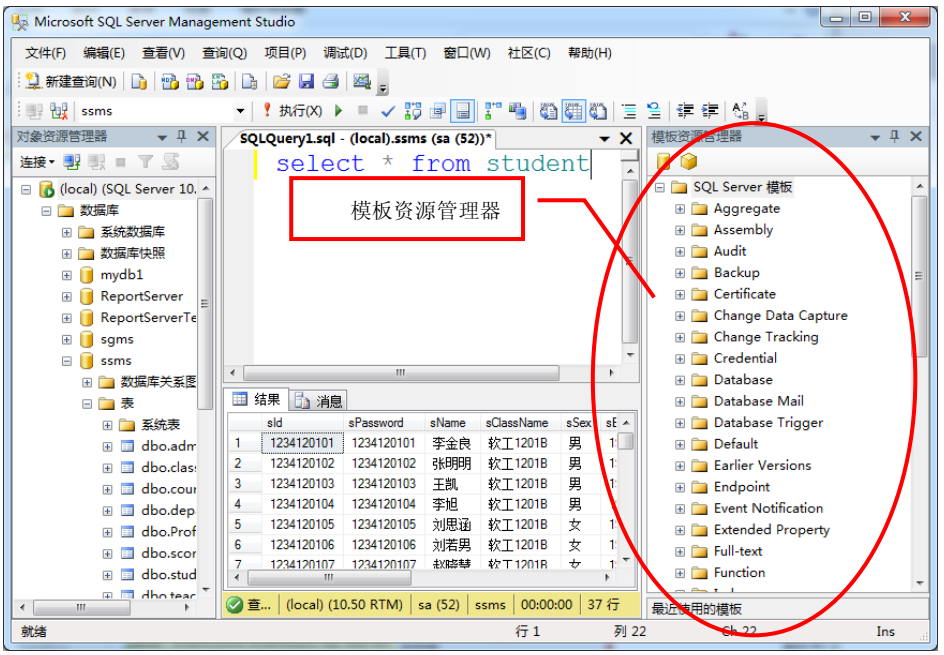


图 2-45 模板对象资源管理器窗口

在模板资源管理器中除了可以找到超过 100 个对象以及 Transact SQL 任务的模板之外，还包括备份和恢复数据库等管理任务。

（2） 解决方案资源管理器。

在 SQL Server Management Studio 中，解决方案资源管理器是用来管理项目方案资源的有效工具。在解决方案资源管理器中，项目可以将一组文件结合在一起作为组进行访问。创建新项目的步骤如下：

第一步：单击菜单栏中的“文件”，在弹出的子菜单中选择“新建”，单击“项目”，选择所要创建的项目类型。类型主要有“SQL Server 脚本”、“Analysis Services 脚本（分析服务脚本）”和“SQL Server Compact Edition 脚本”，然后为创建的项目或方案命名，并选择文件的存储路径，单击“确定”按钮，完成项目的创建过程。

第二步：接下来就可以为该项目创建一个或多个（如果所创建的项目连接的数据库不只一个）数据库连接或者添加已经存在的项目文件，如图 2-46 所示，只需要在“解决方案资源管理器”窗格内的“SQL Server 脚本 1”上右击，在弹出的快捷菜单中选择要添加的项目即可。

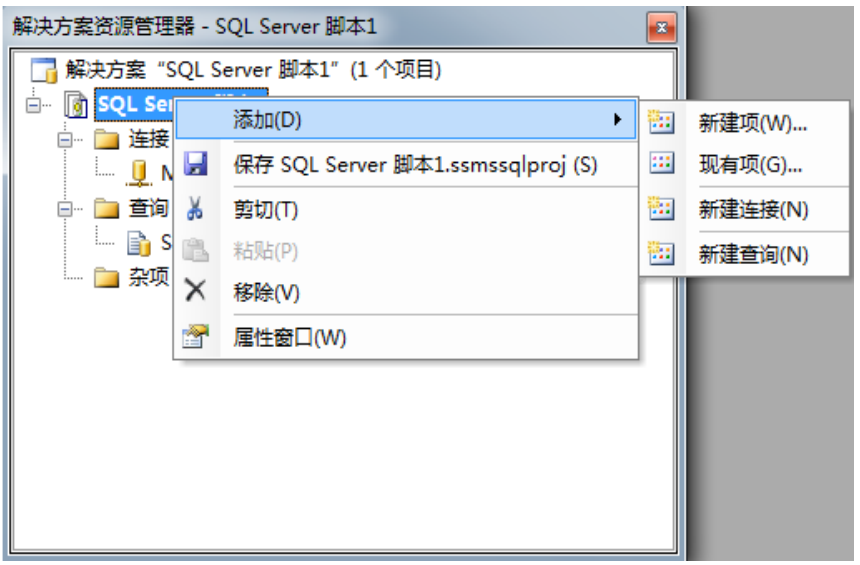


图 2-46　解决方案资源管理器添加项目

三、SQL Server 2008 系统数据库简介

SQL Server 2008 有两类数据库：系统数据库和用户数据库。SQL Server 系统数据库用于存储和管理系统对象的信息，如果系统数据库遭到破坏，那么 SQL Server 将不能正常启动。在安装了 SQL Server 2008 的系统中共将创建 5 个系统数据库：master、model、msdb、tempdb 和 mssqlsystemresource。

系统数据库中保存的系统表用于系统的总体控制，包括系统运行及对用户数据的操作等基本信息，系统数据库分别是 master、model、msdb 和 tempdb。这些数据库的文件存储在 SQL Server 的默认安装目录的 MSSQL 字目录 Data 文件夹中，如图 2-47 所示。

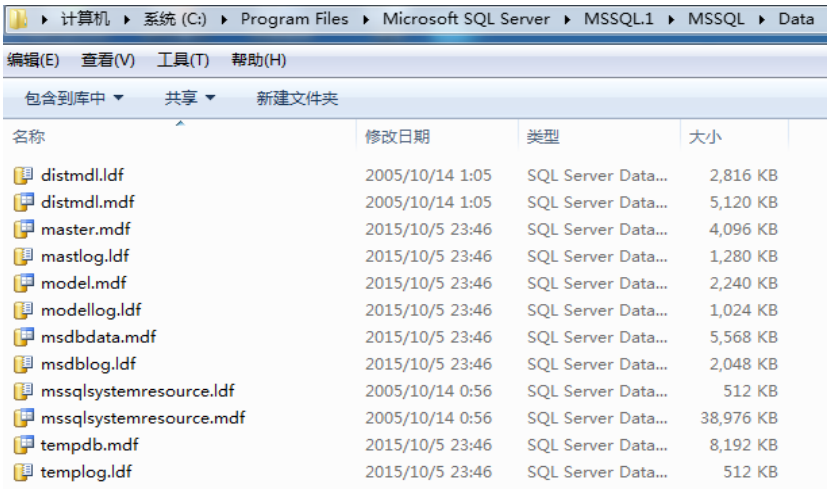


图 2-47　系统表所在目录

1. master 数据库

master 数据库是 SQL Server 中最重要的数据库，它是 SQL Server 的核心数据库，如果该数据库被损坏，SQL Server 将无法正常工作，master 数据库中包含所有的登录名或用户 ID 所属的角色、服务器中的数据库的名称及相关的信息、数据库的位置、SQL Server 如何初始化四个方面的重要信息。

2. model 数据库

model 数据库是 SQL Server 中所有用户数据库和临时数据库(tempdb)的模板数据库。用户创建数据库时是以一套预定义的标准为模型的。例如，若希望所有的数据库都有确定的初始大小，或者都有特定的信息集，那么可以把这些信息放在 model 数据库中，以 model 数据库作为其他数据库的模板数据库。如果想要使用所有的数据库都有一个特定的表，可以把该表放在 model 数据库里，当用户创建一个数据库时，model 数据库的内容会自动复制到该数据库中，但是每个表及其他数据库对象的内容反映的是新建数据库的信息。

3. msdb 数据库

msdb 数据库用于存储作业、报警以及操作员信息。SQL Server 代理服务通过这些信息调度作业和监视数据库系统的错误，并将作业或报警信息传递给操作员。

4. tempdb 数据库

tempdb 数据库用作系统的临时存储空间，其主要作用是存储用户建立的临时表和临时存储过程，以及其他的临时对象或中间结果。存储用户说明的全局变量值，为数据排序创建临时表，存储用户利用游标说明所筛选出来的数据。在执行数据查询时，如果需要创建中间数据交换表来完成，那么中间表就创建在 tempdb 数据库中。SQL Server 每次启动时都重新创建 tempdb 数据库。

5. mssqlsystemresource 数据库

mssqlsystemresource 数据库是一个隐藏的只读数据库，存储系统数据库的可执行对象，

包括 SQL Server 系统的存储过程、函数等。

系统数据库是可以在必要时自动扩展数据库的，这意味着 master、model、msdb、tempdb 和其他关键的数据库将不会在正常的情况下缺少空间。系统数据库具体信息如表 2-4 所示。

表 2-4　SQL Server 系统表

系统数据库	主文件	逻辑名称	物理名称	文件增长比例
master	主数据	master	master.mdf	按 10%自动增长，直到磁盘已满
	日志	masterlog	masterlog.ldf	按 10%自动增长，直到磁盘已满
model	主数据	model	modelr.mdf	按 1MB 自动增长，直到磁盘已满
	日志	modellog	modellog.ldf	按 10%自动增长，直到磁盘已满
msdb	主数据	msdbdata	msdbdata.mdf	按 10%自动增长，直到磁盘已满
	日志	msdblog	msdblog.ldf	按 10%自动增长，直到磁盘已满
tempdb	主数据	tempdb	tempdb.mdf	按 10%自动增长，直到磁盘已满
	日志	templog	templog.ldf	按 10%自动增长，直到磁盘已满

案例 2　安装并熟悉Java Web开发环境

任务 1　JDK的安装与配置

一、下载 JDK

JDK 可以在 Sun 公司的官方网站（http://java.sun.com）下载，如图 2-48 所示。JDK 目前最新版本是 JDK1.8，单击“Downloads”进入下载页面，鼠标向下移动到 All Downloads 链接部分，可以看到 Sun 公司提供的所有软件的下载链接，如图 2-49 图所示。

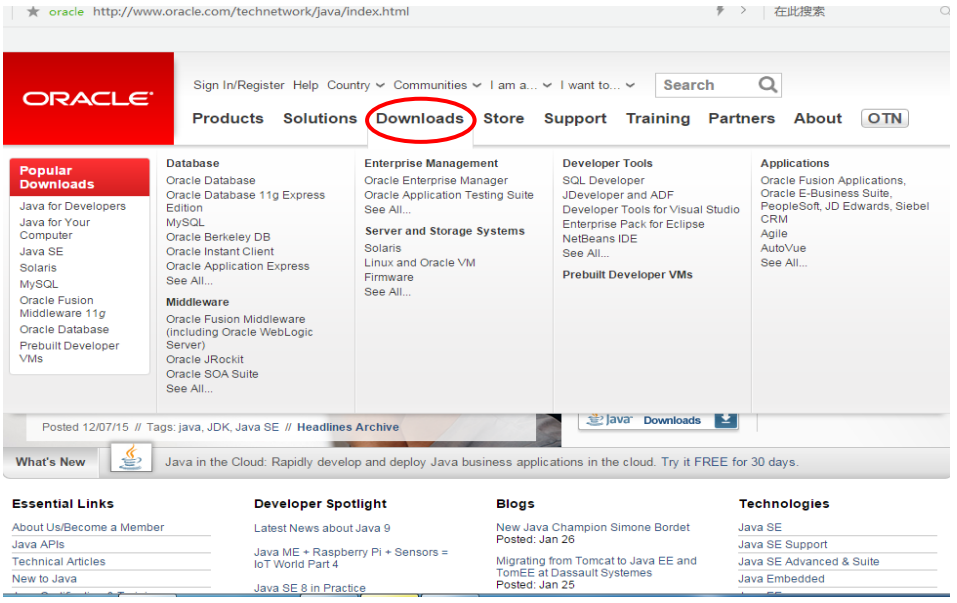


图 2-48　Sun 公司的官方网站首页页面

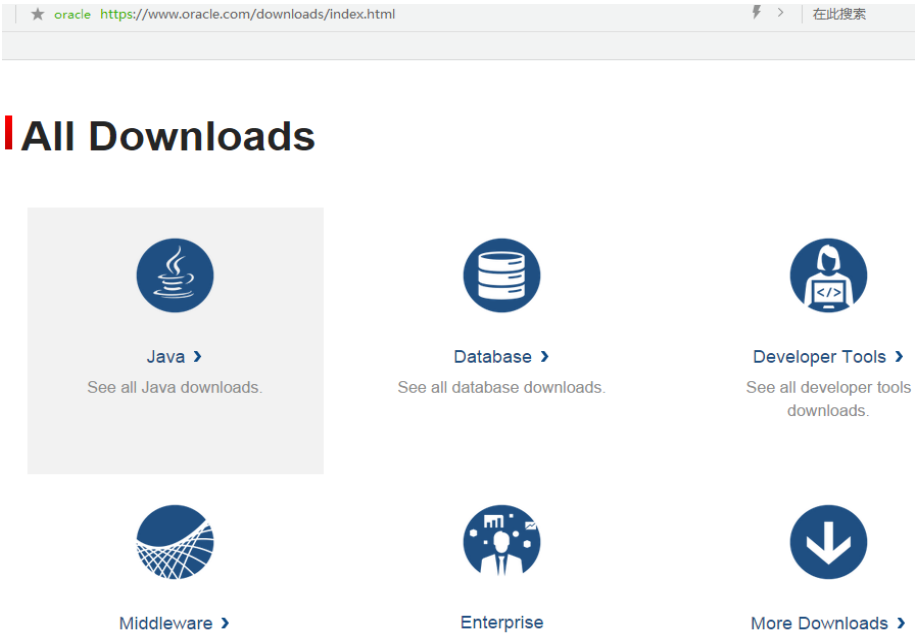


图 2-49　AllDownloads 下载链接页面

在所有下载链接中，单击“Java”进入 Java 相关软件下载页面如图 2-50 所示。

图 2-50　Java 相关软件下载链接

单击其中的 Java SE 进入 Java SE 下载页面如图 2-51 所示。

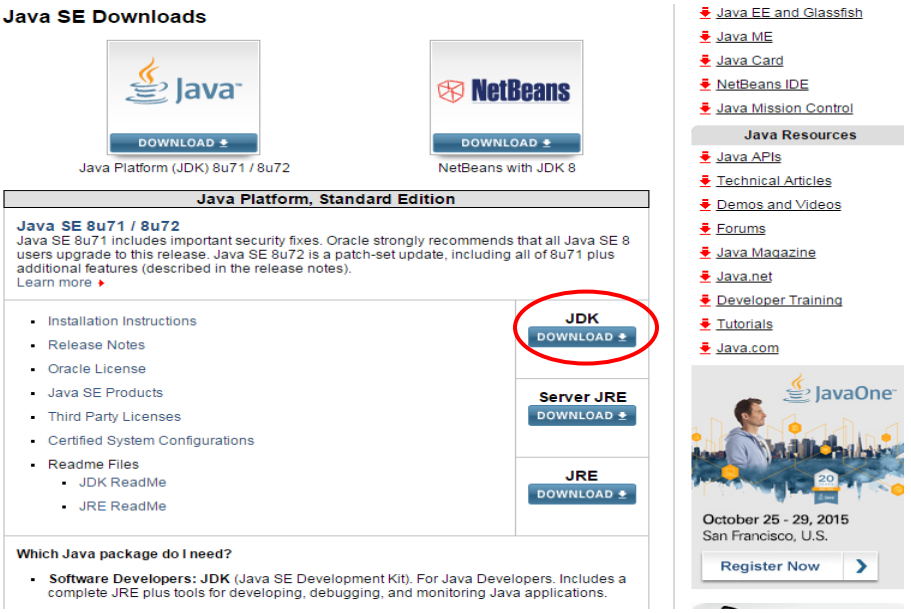


图 2-51　Java SE 下载页

单击 JDK DOWNLOAD 按钮，进入 JDK 下载中心页面如图 2-52 所示，在该页面中提供了最近发布的不同版本的 JDK 的下载链接，根据自己机器系统情况选择相应 32 位或 64 位版本的 JDK。说明：在下载之前首先选中“Accept License Agreement”选项，然后才能下载。

图 2-52　JDK 下载中心页面

二、安装 JDK

JDK 安装包其名称为 jdk-8u71-windows-i586.exe（32 位操作系统使用）或 jdk-8u71-windows-x64.exe（64 位操作系统使用）下载完毕后，就可以在需要开发和运行 Java 程序的机器上安装 JDK 了，其具体步骤如下：

（1） 双击“jdk-8u71-windows-i586.exe”或“jdk-8u71-windows-x64.exe”文件开始安装，安装向导会指导安装过程，如图 2-53 所示。

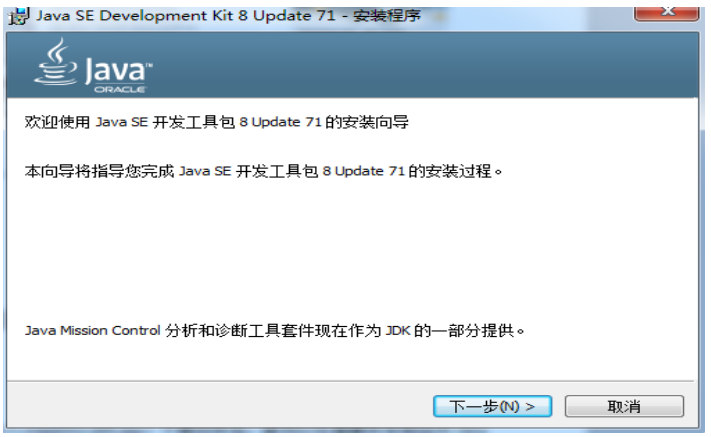


图 2-53　JDK 开始安装页面

（2） 单击“下一步”进入设置 JDK 的安装路径及选择安装组件的窗口，如图 2-54 所示，安装组件选择默认，在该窗口中单击“更改”按钮，可以修改安装路径。

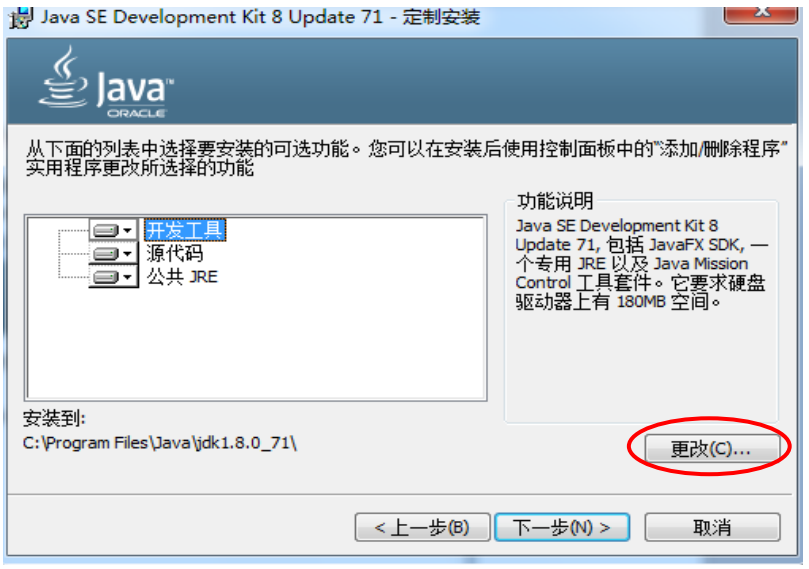


图 2-54　设置 JDK 的安装路径及选择安装组件窗口

（3） 由于 JDK 只是 Java 程序的开发环境，所以在 JDK 的安装文件中还包含了一个

JRE（Java 运行环境），在默认情况下同 JDK 一起安装。JRE 默认和 JDK 安装在同一路径下，如图 2-55 所示。

图 2-55　设置 JRE 的安装路径窗口

在设置 JRE 安装路径的窗口中，单击“更改”按钮，在打开的窗口中将 JRE 的安装路径修改为自己想要的位置。

（4） 单击“下一步”按钮继续安装 JRE。在弹出的安装完成的提示如图 2-56 所示窗口中，单击“关闭”按钮，即可完成 JDK 的安装。

图 2-56　安装完成窗口

三、Windows 系统下 JDK 的配置和测试

JDK 安装完成后，还需要设置环境变量 PAHT 和 CLASSPATH，设置 PATH 是为了在

命令提示符下任何位置都可以运行 JDK 的 java 和 javac 程序，设置环境变量 CLASSPATH 是为了指定 java 类库的位置，设置环境变量的具体步骤如下：

（1） 对 Win7 系统，在桌面的“计算机”图标上右击，选择“属性”菜单项，打开系统属性窗口，如图 2-57 所示。

图 2-57　系统属性设置窗口

在打开的“系统属性”窗口中选择“高级系统设置”选项卡，打开高级系统设置窗口如图 2-58 所示。

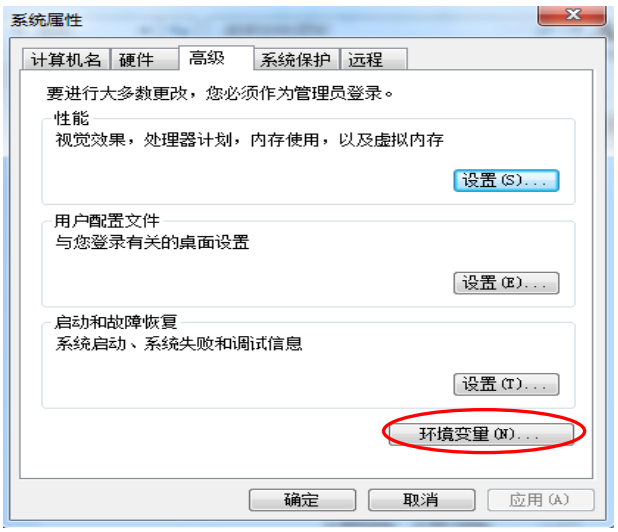


图 2-58　高级系统设置对话框

（2） 单击“环境变量”按钮，打开“环境变量”设置对话框，如图 2-59 所示，在这里添加针对所有用户的“系统变量”（说明：用户变量是只能对当前用户使用的变量，系统变量是系统下任何用户都可以使用的变量）。单击“系统变量”区域中的“新建”按钮，会弹出“新建系统变量”对话框，如图 2-60 所示。

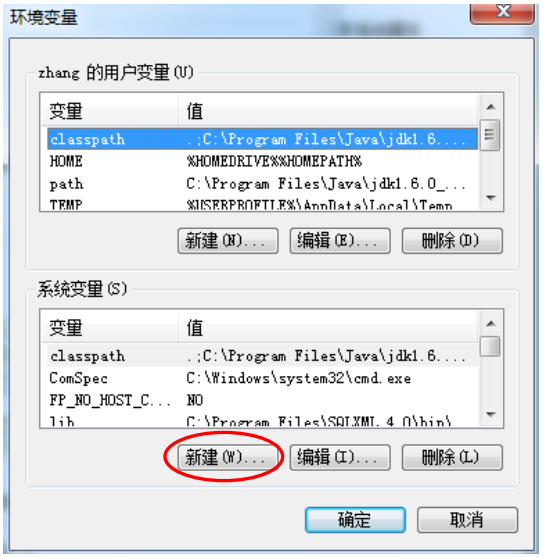


图 2-59　设置“环境变量”对话框

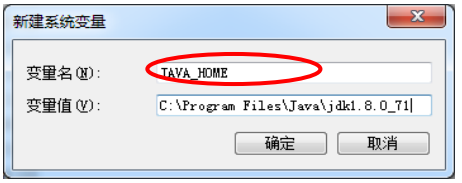


图 2-60　新建系统变量对话框

（3） 在“变量名”文本框中输入“JAVA_HOME”，在“变量值”文本框中输入 JDK 的安装路径“C:\Program Files\Java\jdk1.8.0_71”，单击“确定”按钮，完成环境变量 JAVA_HOME 的配置。

（4） 在系统变量中查看 PATH 变量，如果不存在，则新建变量 PATH，否则选中该变量，单击“编辑”按钮，打开“编辑系统变量”对话框，在该对话框的“变量值”文本框的起始位置添加如下内容“%JAVA_HOME%\bin;”，如图 2-61 所示。

（5） 单击“确定”按钮返回到“环境变量”对话框，在系统变量中查看 CLASSPATH 变量，如果不存在则新建变量 CLASSPATH，变量值为“.;%JAVA_HOEM%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar”，设置如图 2-62 所示。

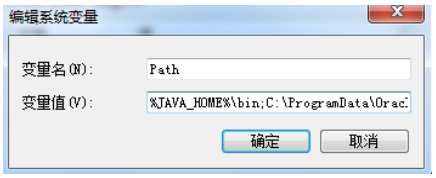


图 2-61　设置 PATH 变量对话框

图 2-62　设置 CLASSPATH 变量对话框

说明：在上面的变量值中，圆点“.”一定不能少，它代表当前路径，否则在当前路径下不能运行.class 文件。%JAVA_HOME%\代表环境变量 JAVA_HOME 的值。

（6） JDK 安装和配置完成后，可以测试 JDK 是否能够在机器上运行。具体步骤是：选择“开始”，在“运行”中输入“cmd”，如图 2-63 所示，打开 dos 命令窗口，在命令提示符下输入 javac 回车，系统会输出 javac 的用法信息，如图 2-64 所示，则 JDK 环境配置成功。如果 JDK 环境配置不成功，则输入 javac 会显示 javac 不是内部或外部命令等提示，

如图 2-65 所示。

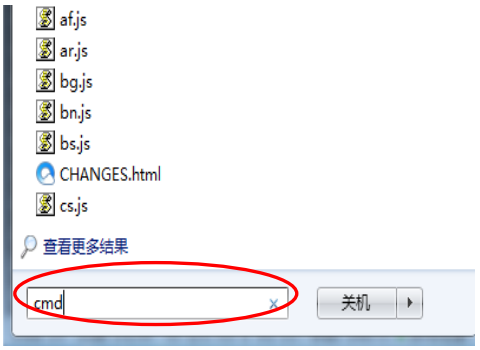


图 2-63　打开 dos 命令窗口

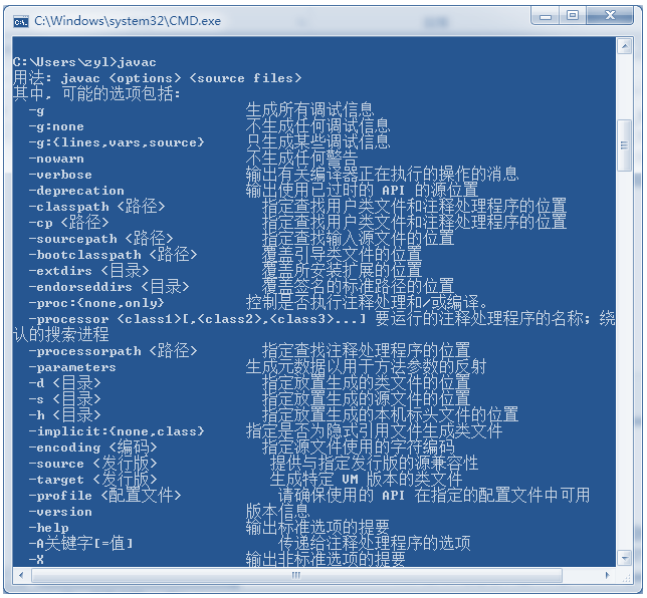


图 2-64　JDK 环境配置成功测试窗口

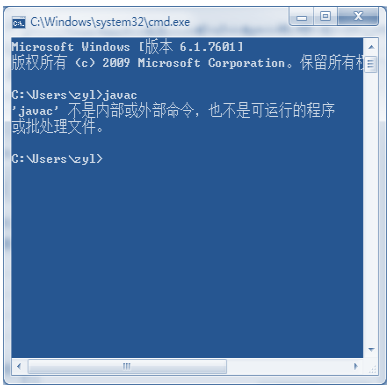


图 2-65　测试 JDK 环境配置不成功窗口

任务 2　Tomcat的安装与配置

一、下载 Tomcat

Tomcat 服务器是 ApacheJakarta 项目组开发的产品，它具有免费和跨平台等诸多特性，目前最新版本是 9.0，Tomcat 服务器作为 Web 服务器的首选。下载 Tomcat 具体步骤如下：

（1） 在 IE 地址栏中输入 http://tomcat.apache.org，进入到 Tomcat 官方网站，如图 2-66 所示。

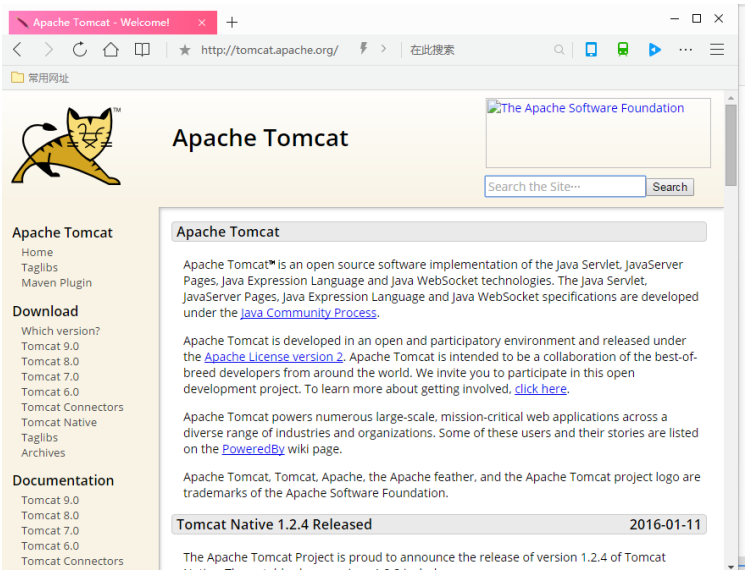


图 2-66　Tomcat 官网首页

（2） 在左侧的 Download 列表中，有 Tomcat 的各种版本，单击“Tomcat 8.0”超链接，进入到 Tomcat 8.0 的下载页面，如图 2-67 所示。

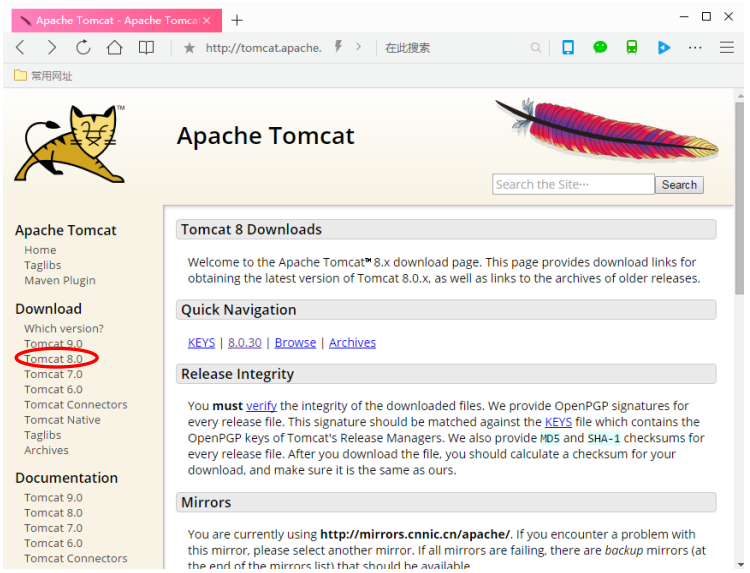


图 2-67　Tomcat 8.0 的下载页面

（3） 单击“Quick Navigation”区域中的“8.0.30”超链接，将直接定位到图 2-68 所示的位置，在此处将显示 Tomcat 8.0.30 的各种下载超链接。

单击 core 节点下的“32-bit/64-bit Windows Service Installer（pgp，md5，sha1）”超链接，将打开“文件下载”对话框，在该对话框中，单击“保存”按钮，即可将 Tomcat 的安装文件下载到本地计算机中。

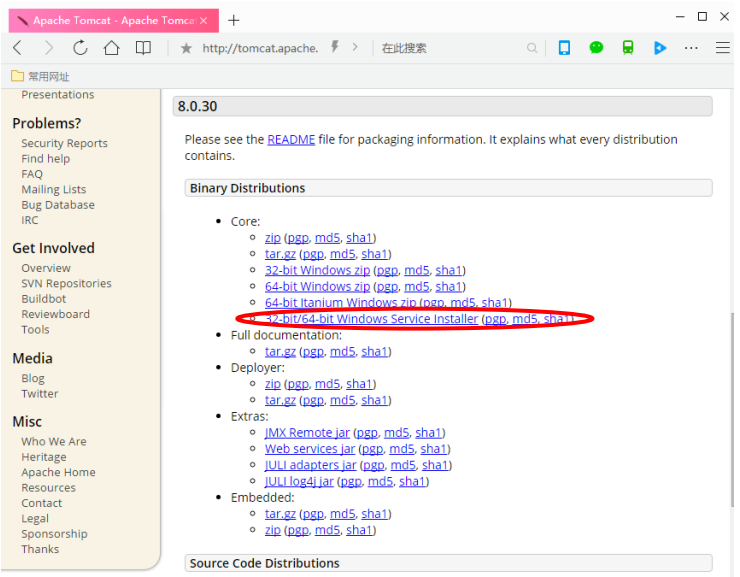


图 2-68　Tomcat 8.0.30 的下载列表

二、安装 Tomcat

Tomcat 安装文件下载完毕后，会得到如图 2-69 所示的文件，通过该文件安装 Tomcat 服务器，具体的安装过程如下：

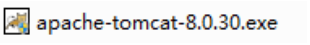


图 2-69　Tomcat 安装文件

（1） 双击“apache-tomcat-8.0.30.exe”文件，打开“安装向导”对话框，单击“Next”按钮后，将打开“许可协议”对话框。

（2） 单击“I Agree”按钮，接受许可协议，将打开“Choose Components”对话框，在该对话框中选择需要安装的组件，通常保留其默认选项，如图 2-70 所示。

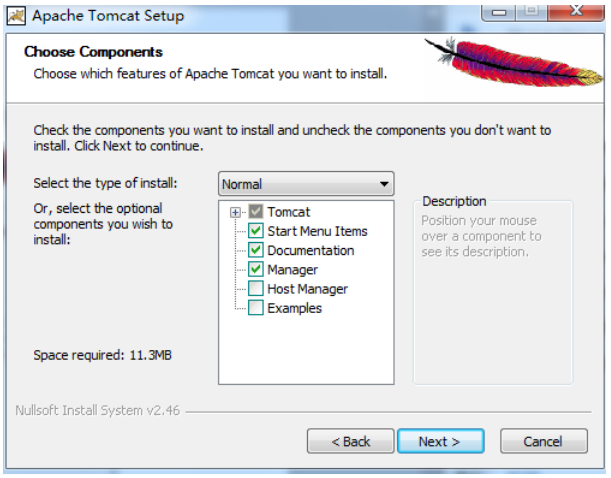


图 2-70　Choose Components 界面

（3） 单击“Next”按钮，在打开的对话框中设置访问 Tomcat 服务器的端口及用户名和密码，通常保留默认配置，即端口号 8080（说明：通常情况下，不要修改默认的端口号，除非 8080 端口已经被占用），用户名 admin，密码为空，如图 2-71 所示。

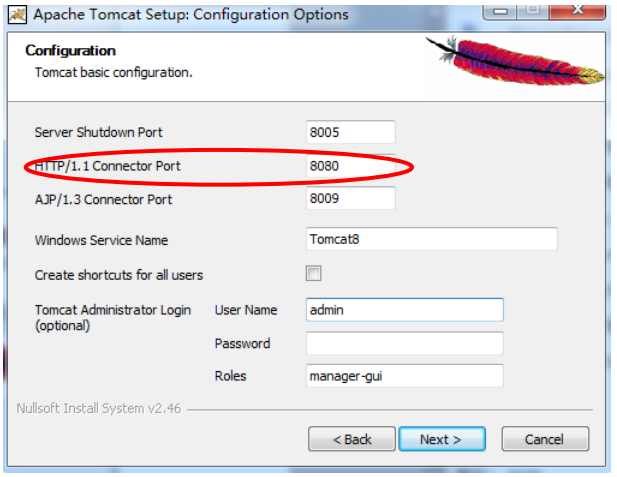


图 2-71　设置 Tomcat 端口号和用户名及密码

（4） 单击“Next”按钮，在打开的“Java Virtual Machine”对话框中选择 Java 虚拟机路径，这里选择 JDK 的安装路径，如图 2-72 所示。

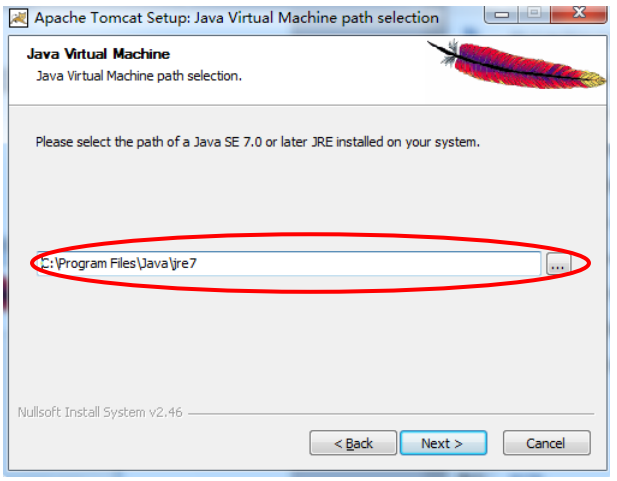


图 2-72　选择 Java 虚拟机路径

（5） 单击“Next”按钮，打开“Choose Install Location”对话框如图 2-73 所示，该对话框中用于选择 Tomcat 的安装路径，单击“Browse”按钮，选择 Tomcat 的安装路径，这里选择默认位置，单击“Install”按钮开始安装 Tomcat。

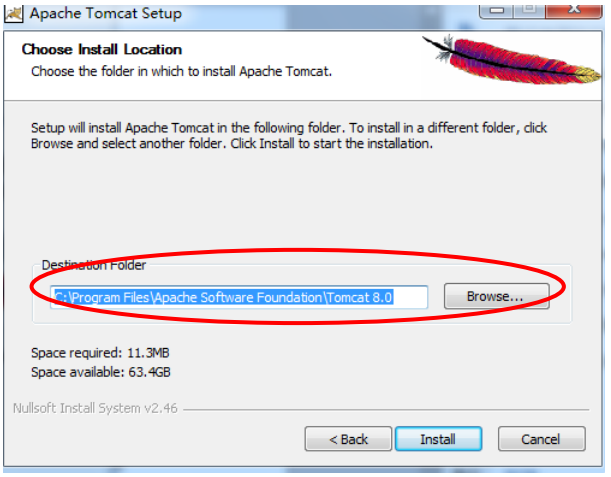


图 2-73　Choose Install Location 对话框

安装过程很快，安装完成会弹出如图 2-74 所示安装完成对话框。取消“Run Apache Tomcat”和“Show Readme”两个复选框的选中，单击“Finish”按钮，即可完成 Tomcat 的安装。

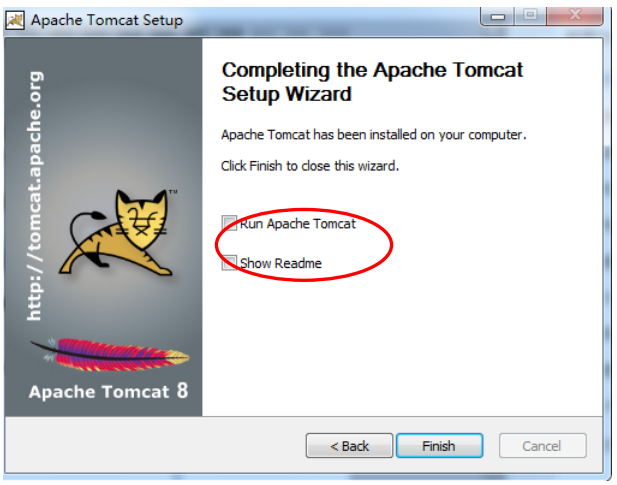


图 2-74　Tomcat 安装完成

三、启动 Tomcat

选择“开始”→“所有程序”→“Apache Tomcat 8.0 Tomcat8”→“Monitor Tomcat”命令，在任务栏右侧的系统托盘中将出现图标，在该图标上右击，在打开的快捷菜单中选择“Start service”菜单项，启动 Tomcat，Tomcat 服务器启动后，在任务栏右侧的系统托盘中的图标变为。

打开 IE 浏览器，在地址栏中输入地址 http://localhost:8080 访问 Tomcat 服务器，若出现如图 2-75 所示页面，则表示 Tomcat 安装成功。

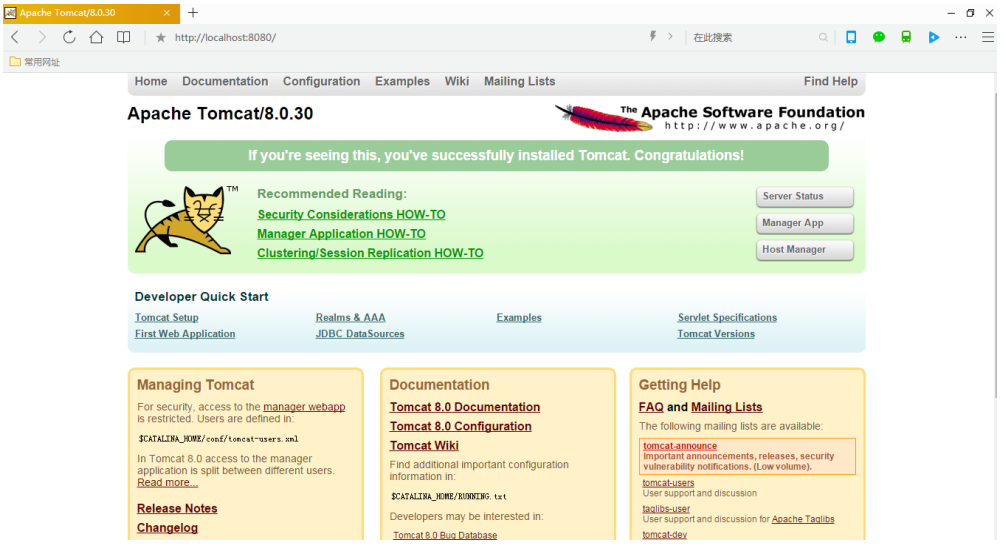


图 2-75　Tomcat 启动页面

四、Tomcat 的目录结构

Tomcat 服务器安装成功后，在 Tomcat 的安装目录下，会出现 7 个文件夹及 4 个文件如图 2-76 所示，下面对这 7 个文件夹的作用分别给以介绍。

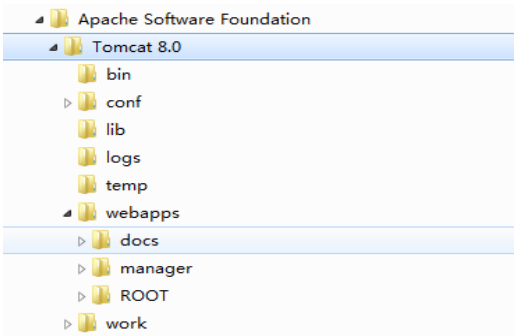


图 2-76　Tomcat 目录结构

（1）　/bin：存放启动与控制 Tomcat 的命令文件的文件夹。

（2）　/conf：存放 Tomcat 服务器的各种全局配置文件，其中最重要的是 server.xml 和 Web.xml。

（3）　/lib：存放 Tomcat 服务器所需的各种 JAR 文件。

（4）　/logs：存放 Tomcat 执行时的日志文件。

（5）　/temp：存放临时文件的文件夹。

（6）　/Webapps：Tomcat 的主要 Web 发布目录，把 Web 应用文件放于此目录下，通过 Tomcat 服务器就可以访问了。

（7）　/work：存放 JSP 编译后产生的 class 文件。

五、修改 Tomcat 的默认端口

Tomcat 默认的服务端口为 8080，但是该端口不是 Tomcat 唯一端口，可以在安装过程中进行修改，如果在安装过程中没有进行修改，还可以通过修改 Tomcat 的配置文件进行修改。下面介绍通过修改 Tomcat 配置文件修改其默认端口的步骤。

（1） 通过任意编辑器如记事本或 editplus 等打开 Tomcat 安装目录下的 conf 文件夹下的 servlet.xml 文件。

（2） 在 servlet.sml 文件中通过查找方式，找到 Connector port="8080"，把 8080 改为要修改的端口号，如 8866，注意：为避免与公共端口号发生冲突，除非默认端口号 8080 被占用，否则一般不修改默认端口号。

（3） 修改成功后，为了使新设置的端口生效，还需要重新启动 Tomcat 服务器。

六、部署 Web 应用

将开发完成的 Java Web 应用程序部署到 Tomcat 服务器上，可以采用静态部署方式或动态部署方式。

1. 基本部署

将 Web 项目文件打包复制到 Webapps 目录，Tomcat 的 Webapps 目录是 Tomcat 默认的应用目录，该目录存放所有的 Web 应用程序，当服务器启动时，会加载所有这个目录下的应用。可以将 JSP 程序打包成一个 war 包放在目录下，服务器会自动解开这个 war 包，并在这个目录下生成一个同名的文件夹。一个 war 包就是有特定格式的 jar 包，它是将一个 Web 程序的所有内容进行压缩得到的。具体如何打包，可以使用许多开发工具的 IDE 环境，如 Eclipse、NetBeans、ant、JBuilder 等。Eclipse 打包方法如下：打开 Eclipse → 选择要打包的项目 → 右击 → 单击 Export（导出）选择 Web → WAR file → Next 选择要打包的工程 → 选择 Browser（选择打包到哪个路径下，默认为当前工程名中），最后单击 Finish 完成打包。把打包的 WAR 文件复制到 Tomcat 的 Webapps 目录下，然后启动 Tomcat 服务器，再打开 IE 浏览器，然后在 IE 浏览器的地址栏中输入 http://服务器 IP（本机可写成 localhost 或 127.0.0.1）：端口号（默认是 8080/应用程序名称的 URL 地址/Web 项目名称（例如：http://localhost:8080/studentScorePro），就可以运行名为 studentScorePro 的 Java Web 项目了。

这种方式通常是用于开发过程中。在真正的项目实施中可能会受到限制：当使用租借的（或他人的）Web 服务器时，该服务器所在的盘符如果存在权限控制，则不能够将 Web 应用程序复制到该盘符中，此时就不能使用这种方式部署项目，只能使用下列方式。

2. 修改配置文件 server.xml

tomcat 安装目录下的 conf 目录用于存放 Tomcat 的配置文件，该目录下的 server.xml 文件用于配置服务器的有关信息。在该文件最后的<Host />元素中通过子元素<Context />配置 Web 应用的物理路径和虚拟路径。在 server.xml 中指定在 Tomcat 的配置文件中，一个 Web 应用就是一个特定的 Context，可以通过在 server.xml 中新建 Context 里部署一个 JSP 应用程序。在 Tomcat 的安装路径下找到 config 文件夹，在该文件夹中找到 server.xml 文件并打开 （**注意**必须以管理员 administrator 身份登录操作系统，否则修改后的 server.xml 文

件无法保存），在 Host 标签内建一个 Context，内容如下。

```
<Context path="/stu"   docBase="f:/book/studentScorePro/WebRoot"  />
```

其中 path 是虚拟路径，指定访问该应用的路径，docBase 是 Web 应用程序的物理路径，即建立的 Web 项目的物理位置。

注意：

（1） 因为 web 项目不要源码部分，只要 WebRoot，所以一定要加上 WebRoot。

（2） 另外，Context 标签的第一个字母是大写的 C。

（3） 该元素应该位于<Host />元素中的最后位置，即</Host>的上面一行。

如果按照上面的配置，则访问该 Web 项目的路径应该是：http://localhost:8080/stu/。

使用这种方式比较方便操作，不需要复制 Web 应用，并且不要求 Web 应用的位置与 Tomcat 服务器在同一个盘符下。

3. 扩展部署

这种方式是在上一种方式的基础上进行了扩展，避免修改 Tomcat 的配置文件。进入下列路径“Tomcat 安装目录\conf\Catalina\localhost”，默认情况下，localhost 目录中有 host-manager.xml 和 manager.xml 两个文件，可以自己定义一个 XML 文件配置要部署的项目，文件名为虚拟路径，按照上面修改 server.xml 的配置，通过路径 http://localhost:8080/stu/ 运行 Web 项目，则要建立的文件名应为 stu.xml，如果有多层路径，则用#间隔，如路径为 http://localhost:8080/a/b/c，则文件名为 a#b#c.xml。在该文件中直接通过<Context />元素的 docBase 属性配置 WEB 应用的物理路径即可，如建立一个名为 studentscore.xml 文件，其内容为：<Context docBase="f:/book/studentScorePro/WebRoot"/>，则访问该 Web 项目的路径应该是：http://localhost:8080/studentscore/。

使用这种方式不需要重启服务器，并且也没有修改 Tomcat 的文件，只是对其进行扩展。在实际的部署中推荐使用这种方式。

任务 3　安装Eclipse并熟悉Eclipse环境

Eclipse 是一个基于 Java 的、开放源码的、可扩展的应用开放平台，它为编程人员提供了一流的 Java 集成开放环境（Integrated Development Environ,ent,IDE）。它是一个可以用于构建集成 Web 和应用程序开发工具的平台，其本身并不会提供大量的功能，而是通过插件来实现程序的快速开发功能。但是，在 Eclipse 的官方网站中提供了一个 JavaEE 版本的 Eclipse IDE。应用 Eclipse IDE for Java EE，可以在不需要安装其他插件的情况下创建动态 Web 项目，可以从 http://www.eclipse.org 网站下载最新的 Eclipse 版本。目前最新版本为 Eclipse4.5.0，它在 Windows 下的安装过程非常简单，只要将下载的 zip 压缩包解压以后就可以直接运行了，但前提是系统已经安装了 JDK。下面将以 Eclipse4.5.0 为例介绍下载 Eclipse 的具体步骤：

一、下载并安装 Eclipse

（1） 在 IE 地址栏中输入 http://www.eclipse.org 进入到 Eclipse 官方网站，如图 2-77 所示：

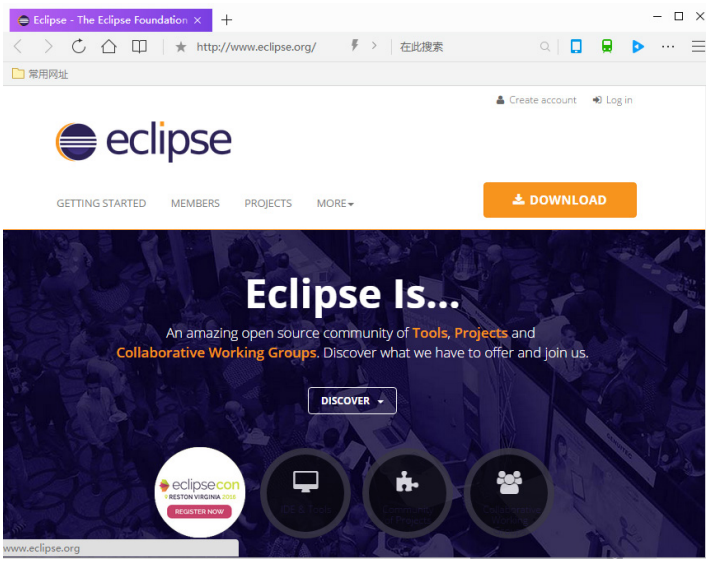


图 2-77　Eclipse 官方网站首页

（2） 单击“Downloads”超级链接，进入到 Eclipse 的下载列表页面，如图 2-78 所示，在该页面中，包括很多 Eclipse IDE 开发工具。

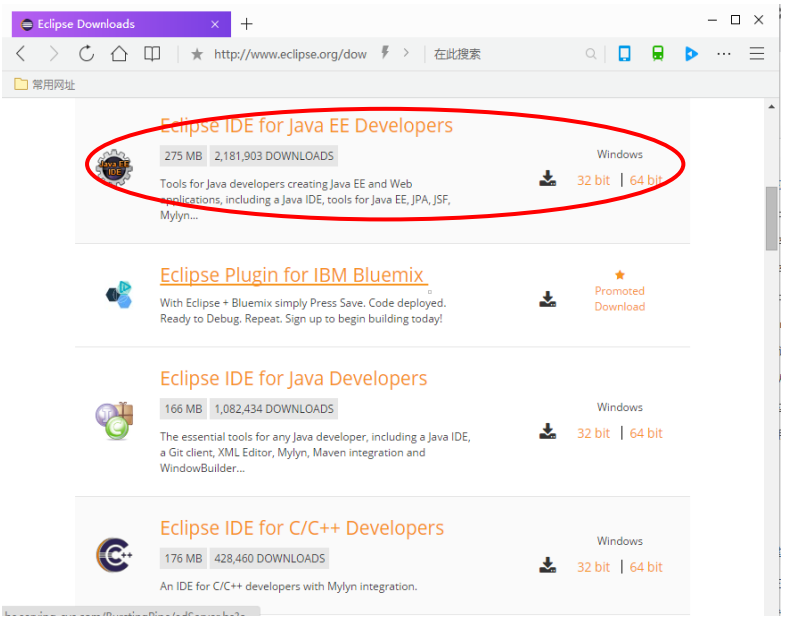

Eclipse Downloads
http://www.eclipse.org/dow
在此搜索
常用网址
Eclipse IDE for Java EE Developers
275 MB
2,181,903 DOWNLOADS
Tools for Java developers creating Java EE and Web
applications, including a Java IDE, tools for Java EE, JPA, JSF,
Mylyn...
Windows
32 bit | 64 bit
Eclipse Plugin for IBM Bluemix
With Eclipse + Bluemix simply Press Save. Code deployed.
Ready to Debug. Repeat. Sign up to begin building today!
Promoted
Download
Eclipse IDE for Java Developers
166 MB
1,082,434 DOWNLOADS
The essential tools for any Java developer, including a Java IDE,
a Git client, XML Editor, Mylyn, Maven integration and
WindowBuilder...
Windows
32 bit | 64 bit
Eclipse IDE for C/C++ Developers
176 MB
428,460 DOWNLOADS
An IDE for C/C++ developers with Mylyn integration.
Windows
32 bit | 64 bit

图 2-78　Eclipse 下载列表页面

（3） 在图 2-78 中找到“Eclipse IDE for Java EE Developers 275M”，根据自己机器系统情况单击其右侧的 Windows 32bit 或 64bit，32 位系统下载 Windows 32bi，64 位系统下载 Windows 64bit。然后会弹出如图 2-79 所示页面，在 choose a mirror close to you Asia 对应的 China 标示的链接中选择一个链接，将打开如图 2-80 所示的文件下载对话框。在该对话框中，单击“下载”按钮，即可将 Eclipse 的安装文件下载到本地计算机中。

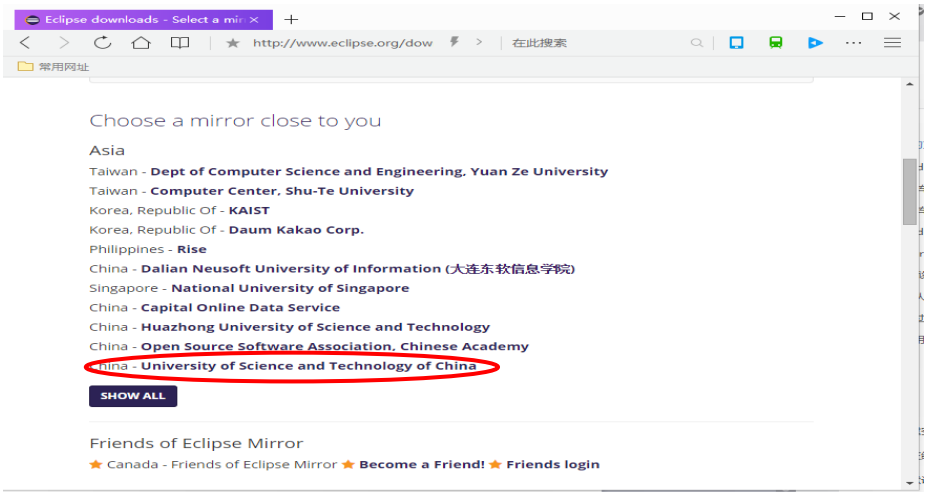

Eclipse downloads - Select a mir
http://www.eclipse.org/dow
在此搜索
常用网址
Choose a mirror close to you
Asia
Taiwan - Dept of Computer Science and Engineering, Yuan Ze University
Taiwan - Computer Center, Shu-Te University
Korea, Republic Of - KAIST
Korea, Republic Of - Daum Kakao Corp.
Philippines - Rise
China - Dalian Neusoft University of Information (大连东软信息学院)
Singapore - National University of Singapore
China - Capital Online Data Service
China - Huazhong University of Science and Technology
China - Open Source Software Association, Chinese Academy
China - University of Science and Technology of China
SHOW ALL
Friends of Eclipse Mirror
Canada - Friends of Eclipse Mirror
Become a Friend!
Friends login

图 2-79　Eclipse 下载列表

图 2-80　文件下载对话框

（4） 下载后的文件名称为 eclipse-jee-mars-1-win32-x86_64.zip。Eclipse 下载完成后，将解压后的文件放置在自己喜欢的路径下，即可完成 Eclipse 的安装（注意 Eclipse 4.5.0 需要 JDK 1.8，如果 JDK 版本过低，不能打开 Eclipse 4.5.0，需要重新安装 JDK1.8）。

二、启动 Eclipse

完成 Eclipse 安装后，就可以启动 Eclipse 了，双击 Eclipse 安装目录下的 eclipse.exe 文件，即可启动 Eclipse，如图 2-81 所示。

图 2-81　Eclipse 启动页面

在初次启动 Eclipse 时，需要设置工作空间，如图 2-82 所示。

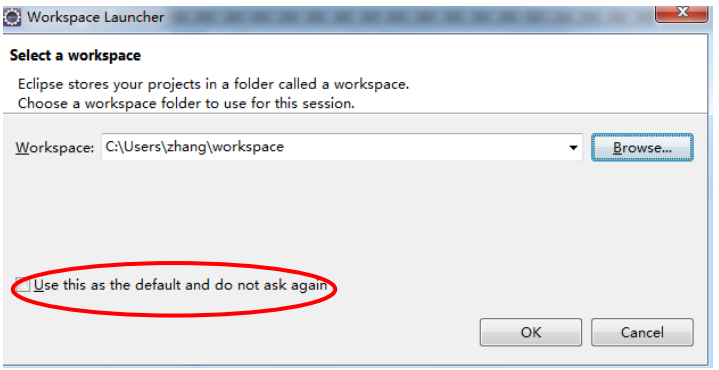


图 2-82　设置工作空间

在每次启动 Eclipse 时，都会弹出设置工作空间的对话框，如果想在以后启动时，不再进行工作空间设置，可以勾选“Use this as the default and do not ask again”复选框，单击“OK”按钮后，即可启动 Eclipse，进入如图 2-83 所示的 Eclipse 起始欢迎页面。

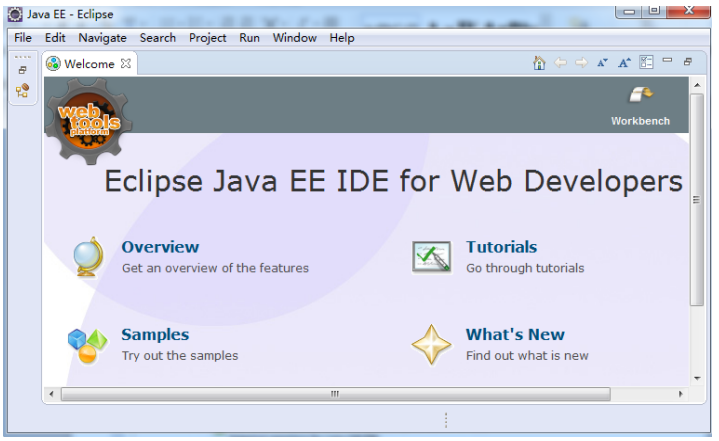


图 2-83　Eclipse 起始欢迎页面

关闭欢迎页面，即打开 Eclipse 的编辑窗口。如图 2-84 所示。

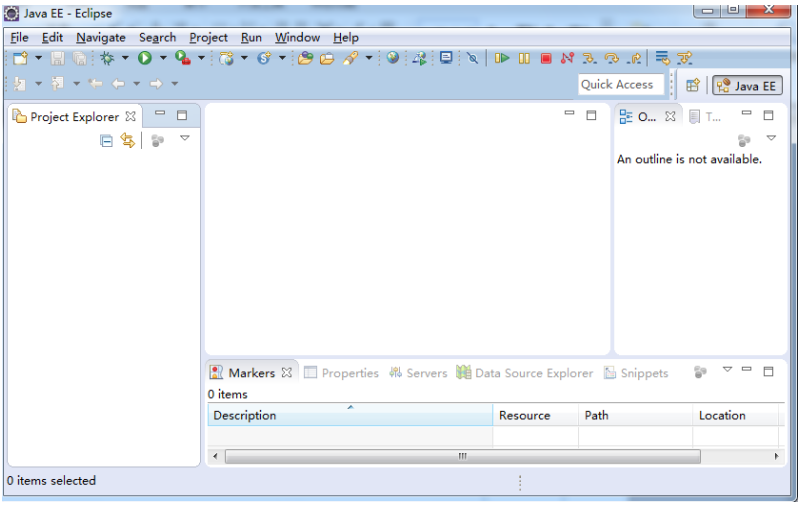


图 2-84　Eclipse 编辑页面

三、安装 Eclipse 中文语言包

直接解压完的 Eclipse 是英文的，通过 Eclipse 的多国语言包，可以根据操作系统的语言环境自动对 Eclipse 进行本地化。具体步骤如下：

在 IE 浏览器的地址栏中输入 http://www.eclipse.org/babel/后，按下回车键将打开 Babel 项目的首页，将光标定位到页面的底部，如图 2-85 所示。

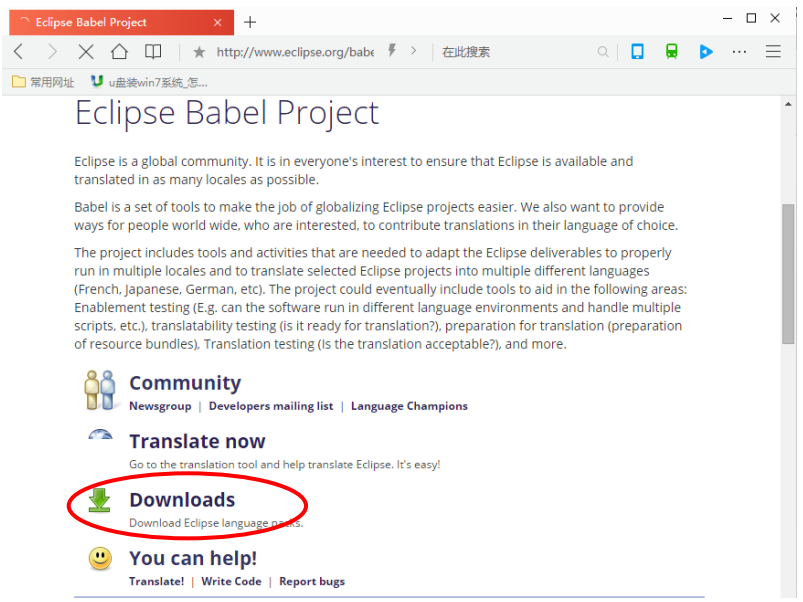


图 2-85　Babel 项目首页

单击 Downloads 会进入 Babel 项目的下载页面，Babel Language Pack Zips and Update

Sites - R0.13.1 有 Mars、Luna、Kepler 三个 Eclipse 版本的链接页面如图 2-86 所示，最新版本的 Mars 语言包还没有出来，如果 Eclipse 为 Mars 版本可以先用 Luna 语言包代替 Mars 语言包，复制如图 2-86 所示的多国语言包得超级链接，以备下面进行在线安装多国语言包。

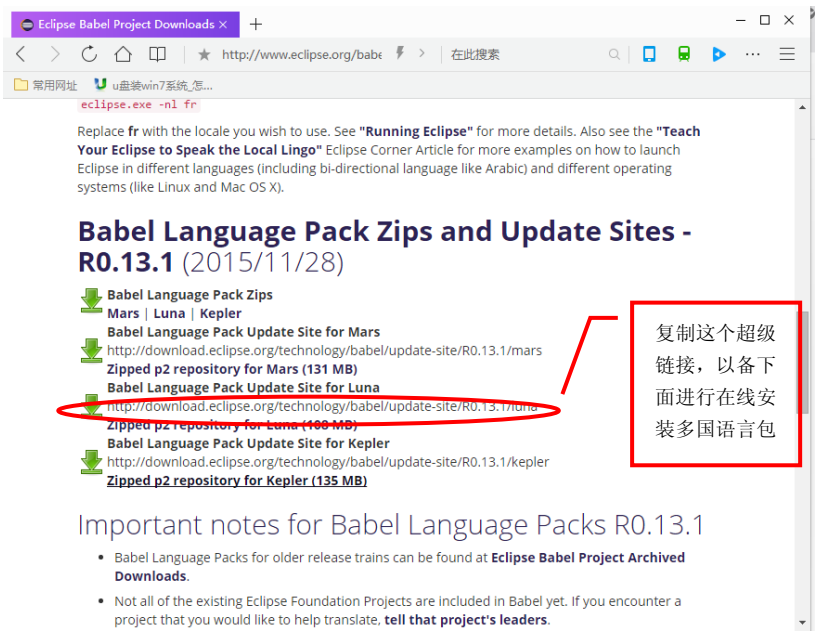


图 2-86　Babel Language Pack Zips and Update Sites 链接页面

这里采用在线安装多国语言包的方法。运行 Eclipse，选择单击 HELP，然后选择 Install New Software 菜单项如图 2-87 所示，会打开如图 2-88 所示的 install 对话框，单击“Add”按钮，会弹出 add Repository 对话框，在该对话框的 Name 文本框中输入“Language”，在 Location 文本框中，粘贴刚刚从 Babel 项目的语言包下载页面复制的 Eclipse4.5 的语言包更新站点的 URL 地址，如图 2-89 所示。

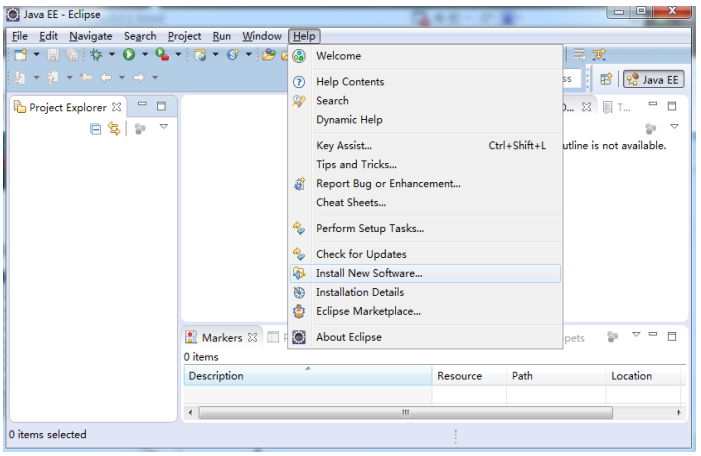


图 2-87　Eclipse 的 Help 菜单

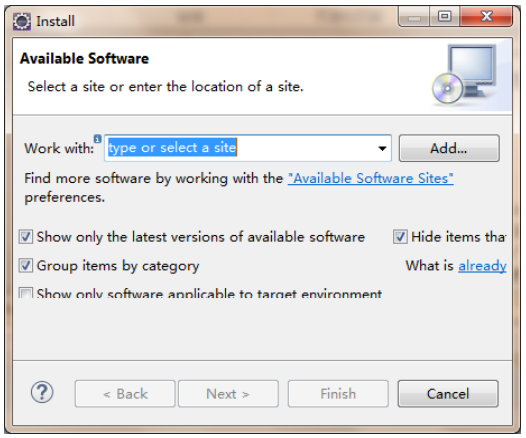

Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
Show only the latest versions of available software
Hide items tha
Group items by category
What is already
< Back
Next >
Finish
Cancel

图 2-88　install 安装语言包窗口

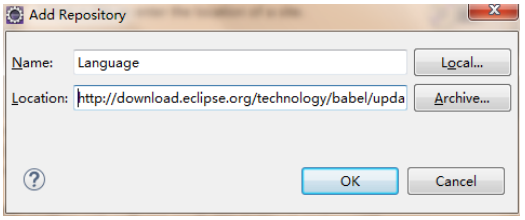


图 2-89　Add Repository 对话框窗口

然后单击“OK”按钮，返回到“install”对话框，在该对话框中将显示 Pending…等待页面如图 2-90 所示，当 Pending 结束后，将显示 Eclipse 4.5 的语言列表，在该列表中找到简体中文的 Eclipse 语言包“Babel Language Packs for eclipse in Chinese（Simplified）”和“Babel Language Packs for eclipse in Chinese(Traditional)”,并选中该语言包前面的复选框。如图 2-91 所示。单击“next”按钮，将显示如图 2-92 所示的查看被安装的项目对话框，在该对话框中显示要安装的项目。

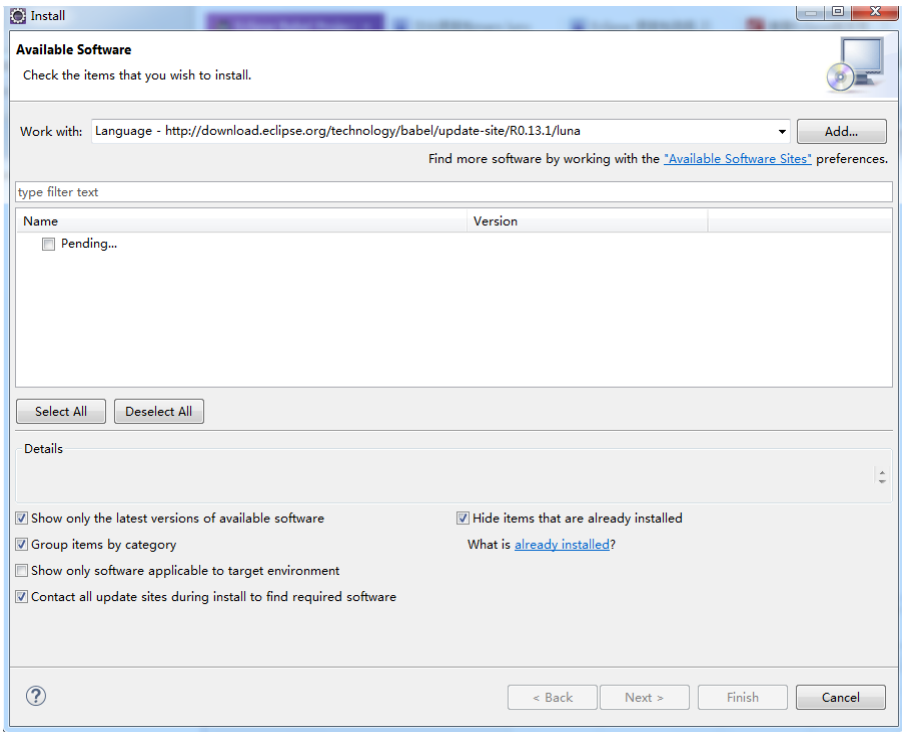


图 2-90　pending 等待页面

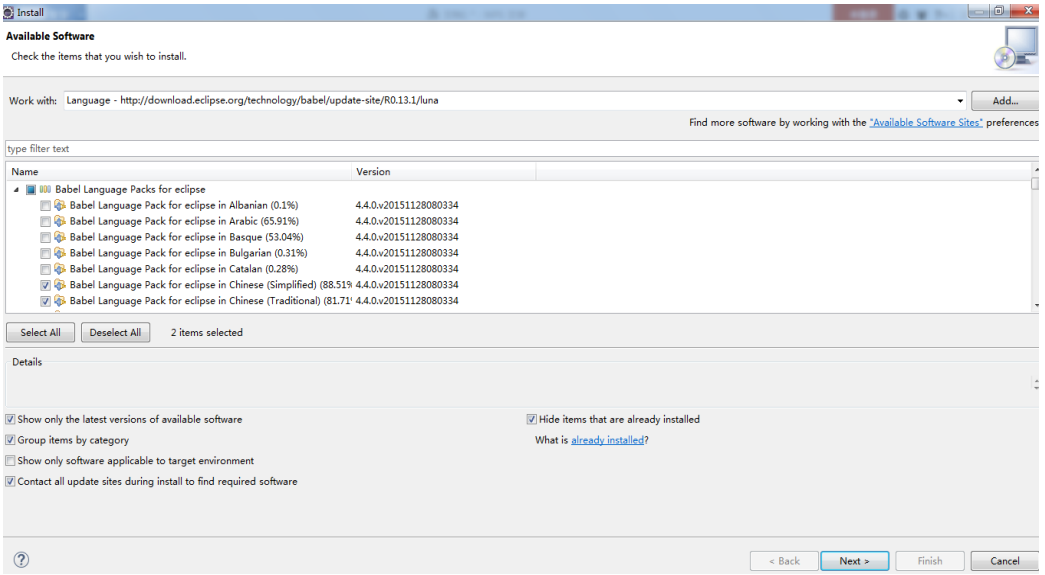


图 2-91　选择安装语言包窗口

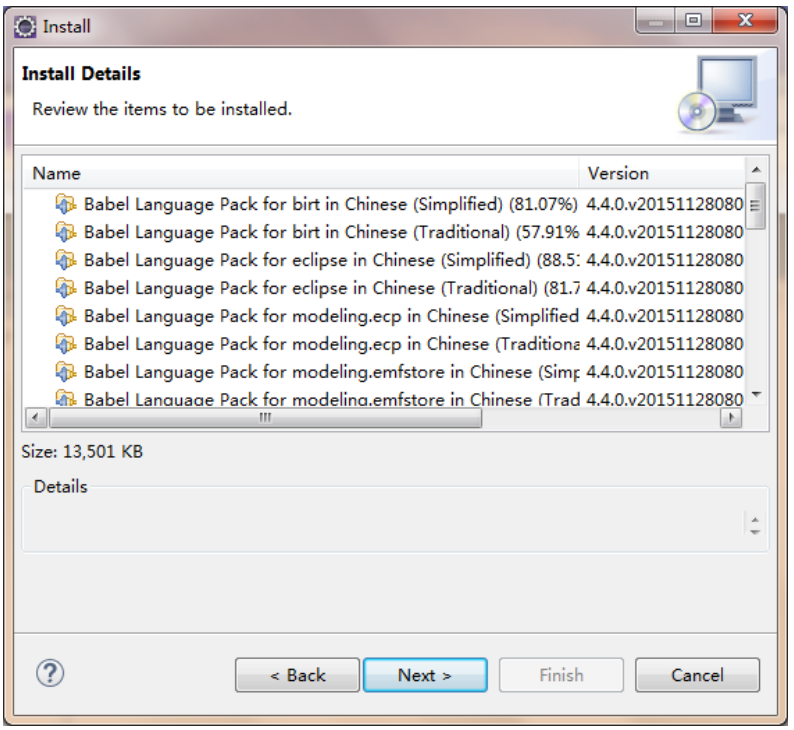


图 2-92　查看被安装的项目对话框

单击“next”按钮，将打开检查许可协议的对话框，在该对话框中，选中“I accept the term of the license argeements”单选按钮，接受许可协议，如图 2-93 所示。

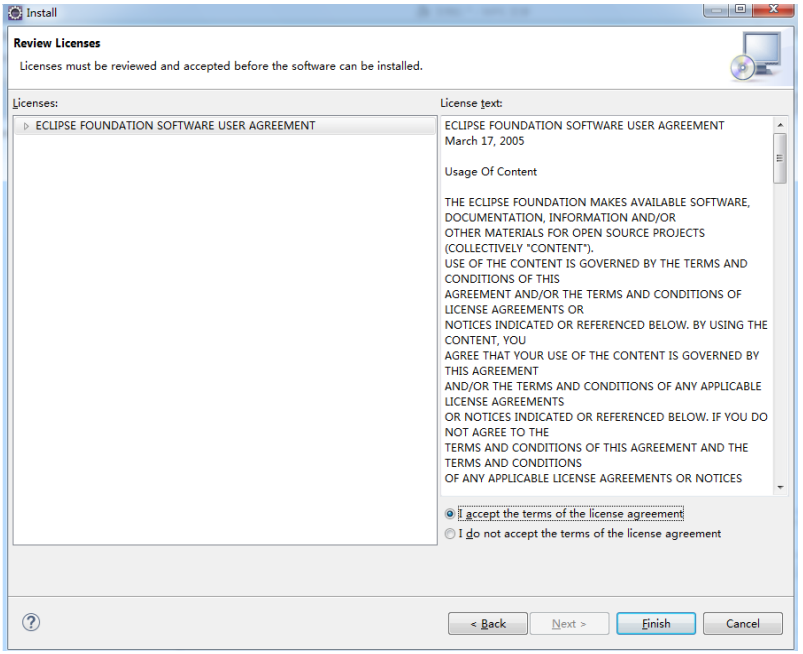


图 2-93　接受许可协议对话框

单击“finish”按钮，将显示安装进度对话框，如图 2-94 所示。因为服务器在国外，这个下载安装的速度会比较慢。在安装过程中会出现安全警告，如图 2-95 所示，这个不用担心安全问题，单击“OK”按钮继续安装。

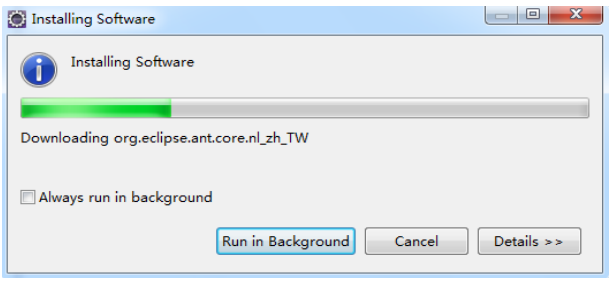


图 2-94　显示安装进度对话框

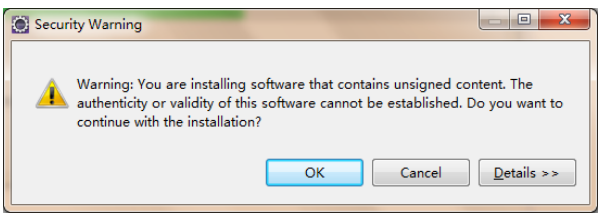


图 2-95　安全警告对话框

安装成功后，会打开询问是否重启 Eclipse 应用多国语言包的对话框，如图 2-96 所示，单击“Yes”按钮，重新启动 Eclipse。重启后的 Eclipse 已经汉化，如图 2-97 所示。如果汉化后启动不了，出现如图 2-98 所示，则需要做以下操作：删除 eclipse/configuration 目录

下的 org.eclipse.osgi 和 org.eclipse.update 两个子目录，重新启动 Eclipse 即可。

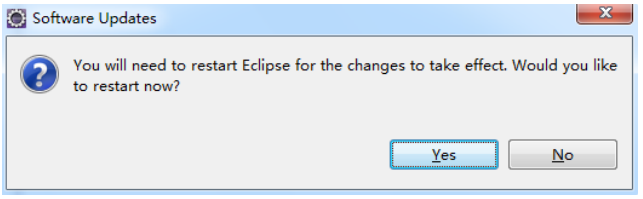


图 2-96　install 对话框的许可协议对话框

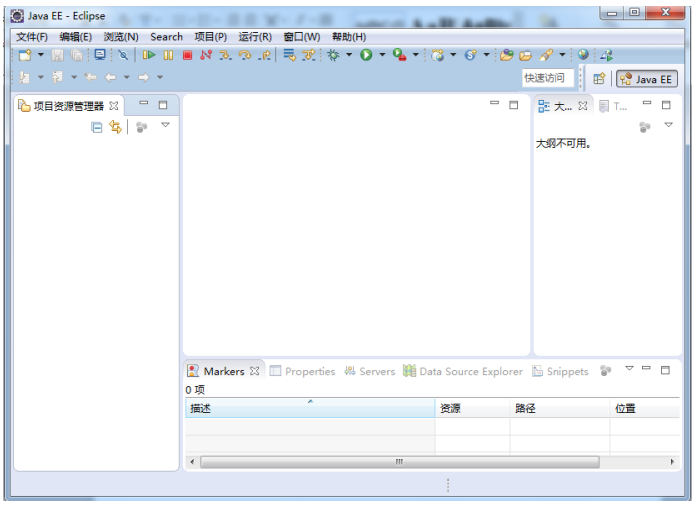


图 2-97　汉化后的 Eclipse

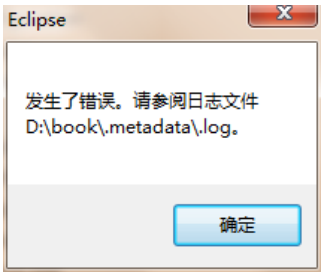


图 2-98　汉化后启动异常信息

相关知识

一、SQL Server 简介

SQL Server 是 Microsoft 公司推出的关系型数据库管理系统，具有使用方便、可伸缩性好与相关软件集成程度高等优点，可跨越从运行 Microsoft Windows 98 的膝上型电脑到运行 Microsoft Windows 2012 的大型多处理器的服务器等多种平台。它最初是由 Microsoft、Sybase 和 Ashton-Tate 三家公司共同开发的，后来 Ashton-Tate 公司退出了该产品的开发，而 Microsoft 公司、Sybase 公司则签署了一项共同开发协议，这两家公司的共同开发结果是发布了用于 Windows NT 操作系统的 SQL Server，1992 年，将 SQL Server 移植到了 Windows NT 平台上。1993 年，SQL Server 4.2 面世，它是一个桌面数据库系统，虽然其功能相对有限，但是采用 Windows GUI，向用户提供了易于使用的用户窗口。

在 SQL Server 4 版本发行以后，Microsoft 公司和 Sybase 公司在 SQL Server 的开发方面分道扬镳，取消了合同，各自开发自己的 SQL Server。Microsoft 公司专注于 Windows NT 平台上的 SQL Server 开发，重写了核心的数据库系统，并于 1995 年发布了 SQL Server 6.05，该版本提供了一个廉价的可以满足众多小型商业应用的数据库方案，而 Sybase 公司则致力于 UNIX 平台上的 SQL Server 的开发。

SQL Server 6.0 版是第一个完全由 Microsoft 公司开发的版本。1996 年，Microsoft 公司推出了 SQL Server 6.5 版本，由于受到旧有结构的限制，微软再次重写 SQL Server 的核心数据库引擎，并于 1998 年发布 SQL Server 7.0，这一版本在数据存储和数据库引擎方面发生了根本性的变化，提供了面向中、小型商业应用数据库功能支持，为了适应技术的发展还包括了一些 Web 功能。此外，微软的开发工具 Visual Studio 6 也对其提供了非常不错的支持。SQL Server 7.0 是该家族第一个得到了广泛应用的成员。

又经过两年的努力开发，2000 年初，微软发布了其第一个企业级数据库系统 SQL Server 2000，其中包括企业版、标准版、开发版、个人版四个版本，同时包括数据库服务、数据分析服务和英语查询三个重要组成。此外，它还提供丰富的管理工具，对开发工具提供全面的支持，对于 Internet 应用提供不错的运行平台，对于 XML 数据也提供了基础的支持。2005 年，五年之后，在整个应用开发环境几乎已经“变了人间”之后，微软发布了新一代数据库产品 SQL Server 2005。

SQL Server 2005 是一个全面的、集成的、端到端的数据解决方案，它为企业中的用户提供了一个安全、可靠和高效的平台用于企业数据管理和商业智能应用。SQL Server 2005 为 IT 专家和信息工作者带来了强大的、熟悉的工具，同时减少了在从移动设备到企业数据系统的多平台上创建、部署、管理及使用企业数据和分析应用程序的复杂度。通过全面的功能集和现有系统的集成性，以及对日常任务的自动化管理能力，SQL Server 2005 为不同规模的企业提供了一个完整的数据解决方案。

2007 年，SQL Server 2008（Katmi）即将发布。

2008 年，SQL Server 2008 于第三季度正式发布，SQL Server 2008 是一个重大的产品版本，它推出了许多新的特性和关键的改进，引入了一批新功能，与早期的相同产品 SQL Server 2005 相比，在存储过程性能方面提供了极大的提升。SQL Server 2008 可以说是目前为止最优秀最完善的 SQL server 数据库。该版本功能可以使用存储和管理许多数据类型，包括 XML、e-mail、时间/日历、文件、文档、地理等，同时提供一个丰富的服务集合来与数据交互作用：搜索、查询、数据分析、报表、数据整合和强大的同步功能。用户可以访问从创建到存档于任何设备的信息，从桌面到移动设备的信息。

2012 年 4 月微软推出了 SQL Server 2012, SQL Server 2012 提供了一个云计算信息平台，该平台可帮助企业对整个组织有突破性的深入了解，并且能够快速在内部和公共云端重部署方案和扩展数据。2014 年微软推出了 SQL Server 2014，SQL Server 2014 带来一套功能强大的核心任务工作负载、智能化业务以及混合云服务。

二、JDK 简介

JDK（Java Development Kit）是 Java 语言的软件开发工具包。有以下三个版本：

（1）SE（J2SE）、standard edition 和标准版三个版本，从 JDK 5.0 开始，改名为 Java SE。

（2） EE（J2EE），enterprise edition，企业版，使用这种 JDK 开发 J2EE 应用程序，从 JDK 5.0 开始，改名为 Java EE。

（3） ME（J2ME），micro edition，主要用于移动设备、嵌入式设备上的 Java 应用程序，从 JDK 5.0 开始，改名为 Java ME。

没有 JDK 的话，无法编译 Java 程序，如果想只运行 Java 程序，要确保已安装相应的 JRE。表 2-5 是各版本的名称及发布日期：

表 2-5　JDK 版本信息表

版　本	名　称	发行日期
JDK 1.1.4	Sparkler（宝石）	1997-09-12
JDK 1.1.5	Pumpkin（南瓜）	1997-12-13
JDK 1.1.6	Abigail（阿比盖尔–女子名）	1998-04-24
JDK 1.1.7	Brutus（布鲁图–古罗马政治家和将军）	1998-09-28
JDK 1.1.8	Chelsea（切尔西–城市名）	1999-04-08
J2SE 1.2	Playground（运动场）	1998-12-04
J2SE 1.2.1	none（无）	1999-03-30
J2SE 1.2.2	Cricket（蟋蟀）	1999-07-08
J2SE 1.3	Kestrel（美洲红隼）	2000-05-08
J2SE 1.3.1	Ladybird（瓢虫）	2001-05-17
J2SE 1.4.0	Merlin（灰背隼）	2002-02-13
J2SE 1.4.1	grasshopper（蚱蜢）	2002-09-16
J2SE 1.4.2	Mantis（螳螂）	2003-06-26
Java SE 5.0 (1.5.0)	Tiger（老虎）	2004-09-30
Java SE 6.0 (1.6.0)	Mustang（野马）	2006-04
Java SE 7.0 (1.7.0)	Dolphin（海豚）	2011-07-28
Java SE 8.0 (1.8.0)	未知	2014-03-18

JDK 包含的基本组件包括：

（1） javac——编译器，将源程序转成字节码。

（2） jar——打包工具，将相关的类文件打包成一个文件。

（3） javadoc——文档生成器，从源码注释中提取文档。

（4） jdb——debugger，查错工具。

（5） java——运行编译后的 Java 程序（.class 后缀的）。

（6） appletviewer：小程序浏览器，一种执行 HTML 文件上的 Java 小程序的 Java 浏览器。

（7） Javah：产生可以调用 Java 过程的 C 过程，或建立能被 Java 程序调用的 C 过程的头文件。

（8） Javap：Java 反汇编器，显示编译类文件中的可访问功能和数据，同时显示字节代码含义。

（9） Jconsole: Java 进行系统调试和监控的工具。

三、Tomcat

1. Tomcat 简介

Tomcat 是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有了 Sun 的参与和支持，最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现，Tomcat 5 支持最新的 Servlet 2.4 和 JSP 2.0 规范。因为 Tomcat 技术先进、性能稳定，而且免费，因而深受 Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的 Web 应用服务器。

Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。当配置正确时，Apache 为 HTML 页面服务，而 Tomcat 实际上运行 JSP 页面和 Servlet。另外，Tomcat 和 IIS 等 Web 服务器一样，具有处理 HTML 页面的功能，另外它还是一个 Servlet 和 JSP 容器，独立的 Servlet 容器是 Tomcat 的默认模式。不过，Tomcat 处理静态 HTML 的能力不如 Apache 服务器，目前 Tomcat 最新版本为 9.0。

2. Tomcat 名称的由来

Tomcat 最初是由 Sun 的软件构架师詹姆斯·邓肯·戴维森开发的。后来他帮助将其变为开源项目，并由 Sun 贡献给 Apache 软件基金会。由于大部分开源项目 O'Reilly 都会出一本相关的书，并且将其封面设计成某个动物的素描，因此他希望将此项目以一个动物的名字命名。因为他希望这种动物能够自己照顾自己，最终，他将其命名为 Tomcat（英语公猫或其他雄性猫科动物）。而 O'Reilly 出版的介绍 Tomcat 的书籍的封面也被设计成了一个公猫的形象，而 Tomcat 的 Logo 兼吉祥物也被设计成了一只公猫。

实训项目

1. 安装并配置 SQL Server 2008 R2，并熟悉 SQL Server Management Studio 环境。
2. 安装并配置 JDK。
3. 安装 Tomcat。
4. 安装 Eclipse IDE for Java EE，并为其安装中文语言包。
5. 在 Eclipse 中创建并发布一个 Web 项目。

模块 3　学生成绩管理系统的设计与实现

学习目标

通过本模块的学习，熟练掌握学生成绩管理系统数据库设计的方法和步骤；掌握 Eclipse 环境下建立和运行 Web 项目的方法；熟悉 JSP 连接 SQL Server 2008 R2 的方法和步骤；熟练掌握使用 select、insert、update、delete 语句对学生成绩管理数据库进行查询、插入、修改和删除操作；熟悉视图的定义和使用；了解数据库的备份和恢复策略。

主要内容

（1） 学生成绩管理系统数据库的设计。
（2） Eclipse 环境下建立和运行 Web 项目的方法。
（3） select、insert、delete、update 语句的使用。
（4） 视图的定义和使用。
（5） JSP 连接 SQL Server 2008 数据库的方法和步骤。
（6） 采用 JSP 技术完成学生成绩管理系统的设计与实现。
（7） 学生成绩管理系统数据库的备份和恢复。

重点与难点

（1） 学生成绩管理系统数据库的设计。
（2） select、insert、delete、update 语句的使用。
（3） 采用 JSP 技术完成学生成绩管理系统的设计与实现。

案例 1　学生成绩管理系统的设计

任务 1　学生成绩管理系统的需求描述

学生成绩管理系统是学生信息管理系统的一个组成部分，其用户有四类：学生、教师、教务员和管理员。通过走访学校师生及深入学校内部调查，对系统需求分析结果如下：

（1） 班级管理功能（教务员）：班级的查找，添加，修改，删除功能。
（2） 专业管理功能（教务员）：专业的查找，添加，修改，删除功能。
（3） 学生管理功能（教务员）：查找，插入，修改，删除某班学生或某个学生的基本

信息，可以一次导入某班全部学生信息。

（4） 课程管理功能（教务员）：课程的查找，添加，修改，删除功能。

（5） 院系管理功能（管理员）：院系的查找，添加，修改，删除功能。

（6） 教师管理功能（管理员）：老师的查找，添加，修改，删除功能。

（7） 往届学生成绩导出功能（管理员）：批量导出往届学生的成绩。

（8） 成绩管理功能（教师）：登记，修改某个学生某门课成绩，可以一次导入某班某门课程的全部学生成绩。

（9） 成绩查询功能（所有用户）：每个学生可以查询自己所有课程的成绩。教师可以查询所带课程的成绩，可以对成绩进行排序和统计。管理员可以查询所有学生所有课程的成绩并进行排序和统计。

（10） 密码修改功能（所有用户）：每个用户都可以修改自己的密码。

任务 2　学生成绩管理系统的概念结构设计

在需求的结果中抽取实体有：学生、课程、班级、专业、院系、老师。

每个实体的属性如图 3-1 至图 3-6 所示。

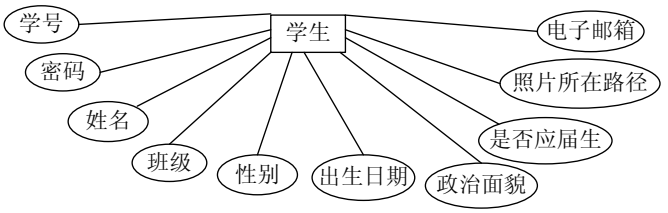


图 3-1　学生实体属性图

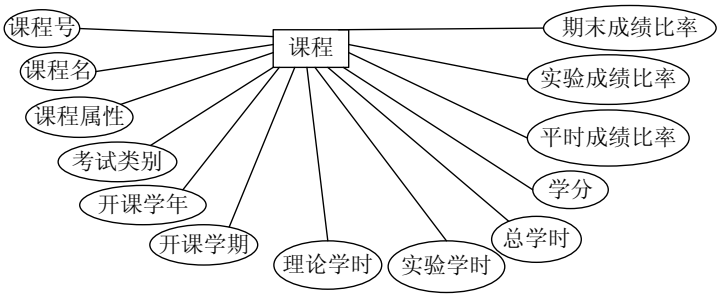


图 3-2　课程实体属性图

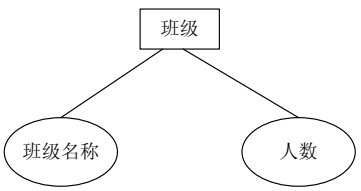


图 3-3　班级实体属性图

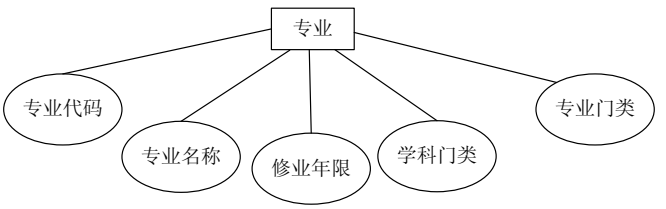


图 3-4　专业实体属性图

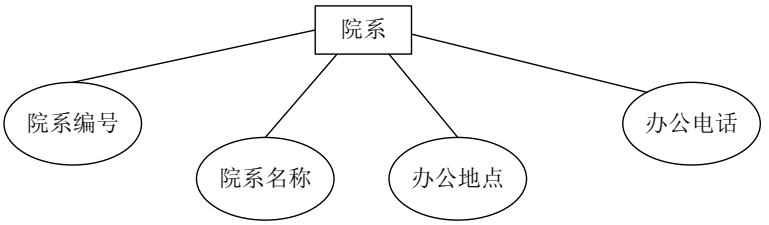


图 3-5　院系实体属性图

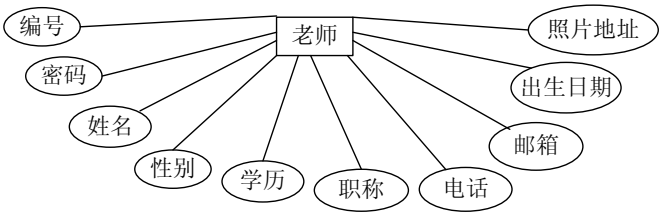


图 3-6　老师实体属性图

一个院系设置多个专业，但同一个学校中，每个院系的专业不能重复；每个专业有多个班级，每个班级有多名学生；每个学生选修多门课程，每门课程由多名学生选修；每门课程由多名教师讲授，每名教师讲授多门课程。

绘制成绩管理子系统的分 E-R 图如图 3-7 和图 3-8 所示：

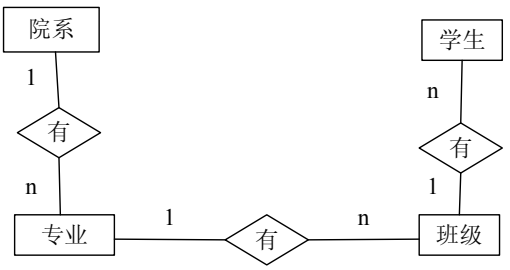


图 3-7　院系－专业－班级－学生实体联系图

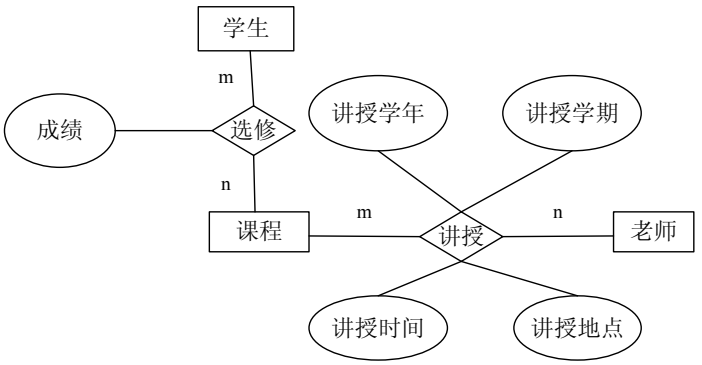


图 3-8　学生－课程－老师实体联系图

合并学生成绩管理系统成绩管理模块整体 E-R 图如图 3-9 所示。

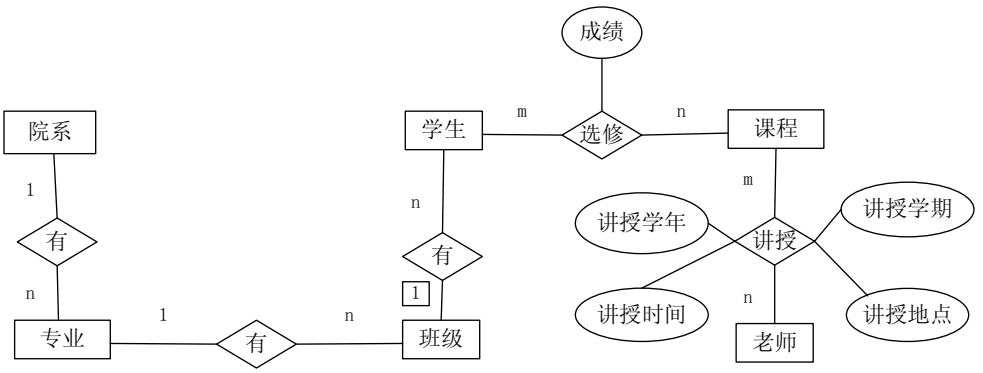


图 3-9　成绩管理模块的 E-R 图

任务 3　学生成绩管理系统的逻辑结构设计

成绩管理模块的 E-R 图转换的关系模式如下：

院系（院系编号，院系名称，办公地点，办公电话）

专业（专业代码，专业名称，修业年限，学科门类，专业门类，院系编号）

班级（班级名称，人数，专业代码）

学生（学号，密码，姓名，性别，出生日期，政治面貌，是否应届生，照片所在路径，电子邮箱，班级名称）

成绩（学年，学期，学号，课程号，平时成绩，考试成绩，实验成绩，总成绩）

课程（课程号，课程名，课程属性，考试类别，开课学年，开课学期，理论学时，实验学时，总学时，学分，平时成绩比率，实验学时比率，期末成绩比率）

讲授（教工编号，课程编号，班级名称，授课时间，授课学年，授课学期，授课地点）

老师（教工编号，密码，姓名，性别，学历，职称，电话，邮箱，出生日期，照片地址）

在关系模式中，带有下划线的属性表示是主码。

为了节省存储空间和避免出现乱码，在数据库系统设计中，字段名通常用英文表示。对应以上关系模式的英文表示如下：

```
    department(did,dname,dOfficePlace,dTelephone)
    Professional(pid,pName,lengthOfSchooling,disciplines,professionalCategor
y,did)
    class(cName,cNumber,pid)
    student(sid,sPassword,sName,sSex,sBirthday,sPolitical,fresh,sPhotoPath,s
Email,sClassName )
    score(academic year,semester,sid,cid,usualResults,examResults,experiment
alResults,totalScore)
    course(cid,cName,cType,cExamtype,cOpenyear,cOpenterm,cTheoryHours,cExper
imentalHours,cTotalHours,cCredit,cUsualResultsRatio,cExperimentalResultsRat
io,cFinalAchievementRatio)
    teaching(tid,cid,classname, tTime,openYear,openTerm,tAddress,)
```

```
   teacher(tid,tPassword,tName,tSex,tEducation,tTitle,tTelephone,tEmail,did,
tBirthday,tPhotopath)
```

任务 4　学生成绩管理系统的物理结构设计

为学生成绩管理数据库 ssms 的数据表建立索引。其中学生表 student 按班级名称 sClassName 升序建立索引，若班级名称相同的再按学生姓名 sName 升序建立索引；课程表 course 按院系编号 pid 升序建立索引，院系编号相同的再按课程名 cname 升序建立索引；成绩表 score 按学号 sid 升序建立索引，若学号相同，则再按课程号 cid 升序建立索引，部门表 department 按部门名称 dname 升序建立索引；专业表 Professional 按院系编号 did 升序建立索引，院系编号相同的再按专业名称 pName 升序建立索引。教师表 teacher 按部门编号 did 升序建立索引，部门编号相同的再按教师姓名 tName 升序建立索引。讲授表 teachering 按教师编号 tid 升序建立索引，教师编号相同的再按课程号 cid 降序建立索引，若课程号相同再按授课班级名升序建立索引。

任务 5　学生成绩管理系统的数据库实现

下面给出成绩管理系统在 SQL Server 2008 下创建数据库的脚本：

```
   create database ssms
   on
   (name='ssms data',
   filename='e:\book\ssms.mdf',
   size=100,
   maxsize=5000,
   filegrowth=10
   )
   log on
   (name='ssms log',
    filename='e:\book\ssms.ldf',
    size=200,
    maxsize=100000,
    filegrowth=10)
    go
   use ssms
    go
    -- ----------------------------
   -- Table structure for  administrator 管理员表
   -- ----------------------------
    create table  administrator (
     aid varchar(30) primary key,
     aPassword  varchar(40) not null,
   aName varchar(8)
    )
    go
   -- Table structure for  Department 院系关系表
   -- ----------------------------
   use ssms
```

```
   go
   create table department
   (did char(2) primary key check (did like '[0-9][0-9]'),--系编号
   dName char(20) unique not null,--系名称
   dOfficePlace char(40) ,--办公地点
   dTelephone char(11) check (dTelephone like'[0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9][0-9][0-9]')--办公电话
   )
    -- ------------------------------
   -- Table structure for  Professional 专业关系表
   -- ------------------------------
   create table Professional(
   pid varchar(7) primary key,--专业代码
   pName char(20) unique not null,--专业名称
   lengthOfSchooling int,--修业年限
   disciplines  char(20), --学科门类
   professionalCategory  char(26),--专业门类
   did char(2) check (did like '[0-9][0-9]'),--所属院系编号
   constraint  fk_professional_department  foreign  key  (did)  references
department(did) on delete cascade  on update cascade
   )
    -- ------------------------------
   -- Table structure for  class 班级关系表
   -- ------------------------------
   create table class(
   cName varchar(30)  primary key ,--班级名程,B 表示本科，ZB 表示专升本
   cNumber int,--人数
   pid varchar(7),--专业编号
   constraint fk_class_professioal foreign key(pid) references professional
(pid) on delete cascade  on update cascade
   )
    -- ------------------------------
   -- Table structure for  student 学生表
   -- ------------------------------
   create table student(
     sid char(10) primary key  check (sid like '[0-9][0-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9][0-9]'),
     --学号（用户名），学号由位数字组成，-2 位代表入学年份，
     --3-4 位代表院系，-8 位代表班级序号，-10 是学生在班级中的序号。
     sPassword varchar(20) not null,--密码
     sName varchar(10) not null,--姓名
     sClassName varchar(30) not null --班级名称
     sSex varchar(2)  default'男' check (sSex in ('男','女')),--性别
     sBirthday datetime default null,--出生日期
     sPolitical varchar(10) default null,--政治面貌
     fresh bit DEFAULT 1 ,            --是否是应届生
     sPhotoPath varchar(255) default null,--照片所在路径
     sEmail varchar(30) default null,--电子邮箱
     constraint fk_student_class foreign key (sClassName) references class
(cName) on delete cascade on update cascade
   )
```

```
go
-- ----------------------------
-- Table structure for  course  课程表
-- ----------------------------
create table course(
 cid varchar(10) primary key,--课程编号
  cName varchar(60) unique ,--课程名
  cType varchar(10) default null,--课程属性
  cExamtype varchar(10),--考试类别
  cOpenyear varchar(20), --开课学年
  cOpenterm varchar(10),--开课学期
  cTheoryHours int,--理论学时
  cExperimentalHours int,----实验学时
  cTotalHours int ,--总学时
  cCredit int default null,--学分
  cUsualResultsRatio  numeric(3,2) null,--平时成绩比率
  cExperimentalResultsRatio numeric(3,2) null,--实验成绩比率
  cFinalAchievementRatio numeric(3,2) null--期末成绩比率
  )
  go
 -- ----------------------------
-- Table structure for  score 成绩表
-- ----------------------------
create table  score(
  academic_year varchar(20) default null,--学年
  semester varchar(10) default null,--学期
  sid char(10) ,--学号
  cid  varchar(10) ,--课程号
  usualResults  int default null,--平时成绩
  examResults int  default null L,--考试成绩
  experimentalResults  int  default null,--实验成绩
  totalScore int default null,--总成绩
  constraint pk_score primary key (sid,cid),
 constraint  fk_t_score_stu foreign key (sid) references student (sid),
  constraint  fk_t_score_course  foreign key(cid) references course (cid)
)
 -- ----------------------------
-- Table structure for  teacher 教师表
-- ----------------------------
create table teacher(
  tid varchar(8) primary key check (tid like '[0-9][0-9][0-9][0-9][0-9]
[0-9][0-9][0-9]'),--教师编号
  tPassword varchar(20) not null,--密码
  tName  varchar(10) not null,--教师姓名
  tSex  varchar(2)  check (tSex in ('男','女')) default '男',--教师性别
 tEducation  varchar(10)   check (tEducation in ('本科','硕士研究生','博士研
究生','专科')) ,--学历
   tTitle  varchar(10) default null,--职称
   tTelephone  varchar(11) default null check (tTelephone like '1[0-9][0-9]
[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),--电话
```

```
      tEmail  varchar(20) default null,--邮箱
      did  char(2) ,--所在系编号
      tBirthday  datetime default null,--出生日期
      tPhotoPath  varchar(255) default null,--照片地址
      constraint fk_teacher_department foreign key (did) references  department
(did)
       )
   -- ----------------------------
   -- Table structure for teaching 授课关系表
   -- ----------------------------
   create table teaching (
   tid varchar(8) ,--教师编号
     cid varchar(10) ,--课程编号
   classname varchar(30),--课程名
   ttime varchar(30) default null,--授课时间
     openYear varchar(20) ,--授课学年
     openTerm varchar(10),--授课学期
     taddress varchar(30) default null,--授课地点
     constraint pk_teaching primary key(tid,cid, classname,ttime),
   constraint   fk_teaching_teacher  foreign   key(tid)  references   teacher
(tid),
   constraint fk_teaching_course foreign  key (cid) references  course(cid),
   constraint fk_teaching_class foreign key(classname) references class(cname)
   )
```

任务 6　修改数据库中表结构

【例 1】修改学生表中班级名称列，将其非空约束改为允许为空。

```
alter table student alter column sClassName varchar(30) null;
```

【例 2】修改班级表，添加辅导员列，其数据类型为 char 类型，长度为 20 位。

```
alter table class add monitor char(20);
```

【例 3】将班级表中新添加的辅导员列的数据类型修改为 varchar 类型，长度改为 18 位。

```
alter table class alter column monitor varchar(18);
```

【例 4】调用系统存储过程 sp_rename 将班级表中辅导员列“monitor”重命名为“instructor”。

```
exec sp_rename 'class.monitor','instructor';
```

注意：在执行该语句的过程中，将出现提示信息“更改对象名的任一部分都可能会破坏脚本和存储过程”。

【例 5】为班级表的辅导员列新建默认值约束“李勇”。

```
alter table class add constraint dv1 default '李勇' for instructor;
```

【例 6】删除为班级表的辅导员列所建的默认值约束。

```
alter table class drop constraint dv1;
```

【例 7】为学生表的性别属性添加取值范围约束为“男”或者“女”。

```
alter table student add constraint dv2 check(sSex in ('男','女'));
```

【例 8】删除为学生表所创建的约束 dv2。

```
alter table student drop constraint dv2;
```

【例 9】删除为班级表所添加的辅导员列。

```
alter table class drop column instructor;
```

习题

1. 分别按如下要求完成表结构的修改。

（1） 在学生成绩管理系统 ssms 数据库中的学生（Student）表添加 Telehpone 属性列，要求数据类型为 11 个长度的 char（定长字符型）。

（2） 更改 ssms 数据库中学生（Student）表的 sPolitical 属性列为 16 个长度的 varchar（可变长度的字符型）。

（3） 为 ssms 数据库中学生成绩表 score 的成绩 examResults 属性列建立用户自定义约束，约束名为 ck_examResults，要求成绩必须在 0 到 100 的范围内。

（4） 为 ssms 数据库中学生（student）表的学号（sId）列建立用户自定义约束，要求学号必须由 10 位阿拉伯数字组成。

（5） 删除 ssms 数据库中学生成绩表 score 在 examResults 属性列上的 check 约束 ck_examResults。

（6） 删除第 1 题中添加的 Telehpone 属性列。

（7） 为 ssms 数据库中学生（Student）表的 Ssex 列添加默认值约束，设置其默认值为“男”，默认值约束的名字为 Stu_sex_default。

2. 数据库设计的基本步骤是什么？每一步的主要任务是什么？

实训项目

通过走访学生、楼管和相关工作人员完成学生宿舍管理系统数据库的设计工作。

相关知识

数据库设计

数据库设计是指根据用户的需求，在某一具体的数据库管理系统上，设计数据库的结构和建立数据库的过程。数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效的运行环境。数据库设计的特点之一是“三分技术，七分管理，十二分基础数据”。

数据库设计是建立数据库及其应用系统的技术，是信息系统开发和建设中的核心技

术。由于数据库应用系统的复杂性，为了支持相关程序运行，数据库设计就变得异常复杂，因此最佳设计不可能一蹴而就，而只能是一个“反复探寻，逐步求精”的过程，也就是规划和结构化数据库中的数据对象以及这些数据对象之间关系的过程。

按照结构化系统设计的方法，考虑数据库及其应用系统开发全过程，将数据库设计分为六个步骤：需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施、数据库运行和维护。

1. 需求分析

调查和分析用户的业务活动和数据的使用情况，弄清所有数据的种类、范围、数量以及它们在业务活动中交流的情况，确定用户对数据库系统的使用要求和各种约束条件等，形成用户需求规约。

2. 概念设计

对用户要求描述的现实世界（可能是一个工厂、一个商场或者一个学校等），通过对其中信息的分类、聚集和概括，建立抽象的概念数据模型。这个概念模型应反映现实世界各部门的信息结构、信息流动情况、信息间的互相制约关系以及各部门对信息储存、查询和加工的要求等。所建立的模型应避开数据库在计算机上的具体实现细节，用一种抽象的形式表示出来。以扩充的实体——（E-R 模型）联系模型方法为例，第一步先明确现实世界各部门所含的各种实体及其属性、实体间的联系以及对信息的制约条件等，从而给出各部门内所用信息的局部描述（在数据库中称为用户的局部视图）。第二步再将前面得到的多个用户的局部视图集成为一个全局视图，即用户要描述的现实世界的概念数据模型。

3. 逻辑设计

主要工作是将现实世界的概念数据模型设计成数据库的一种逻辑模式，即适应于某种特定数据库管理系统所支持的逻辑数据模式。与此同时，可能还需为各种数据处理应用领域产生相应的逻辑子模式。这一步设计的结果就是所谓的“逻辑数据库”。

4. 物理设计

根据特定数据库管理系统所提供的多种存储结构和存取方法等依赖于具体计算机结构的各项物理设计措施，对具体的应用任务选定最合适的物理存储结构（包括文件类型、索引结构和数据的存放次序与位逻辑等）、存取方法和存取路径等。这一步设计的结果就是所谓的“物理数据库”。

5. 验证设计

在上述设计的基础上，收集数据并具体建立一个数据库，运行一些典型的应用任务来验证数据库设计的正确性和合理性。一般情况下一个大型数据库的设计过程往往需要经过多次循环反复。当设计的某步发现问题时，可能就需要返回到前面去进行修改。因此，在做上述数据库设计时就应考虑到今后修改设计的可能性和方便性。

6. 运行与维护

在数据库系统正式投入运行的过程中，必须不断地对其进行调整与修改，如图 3-10 所示。

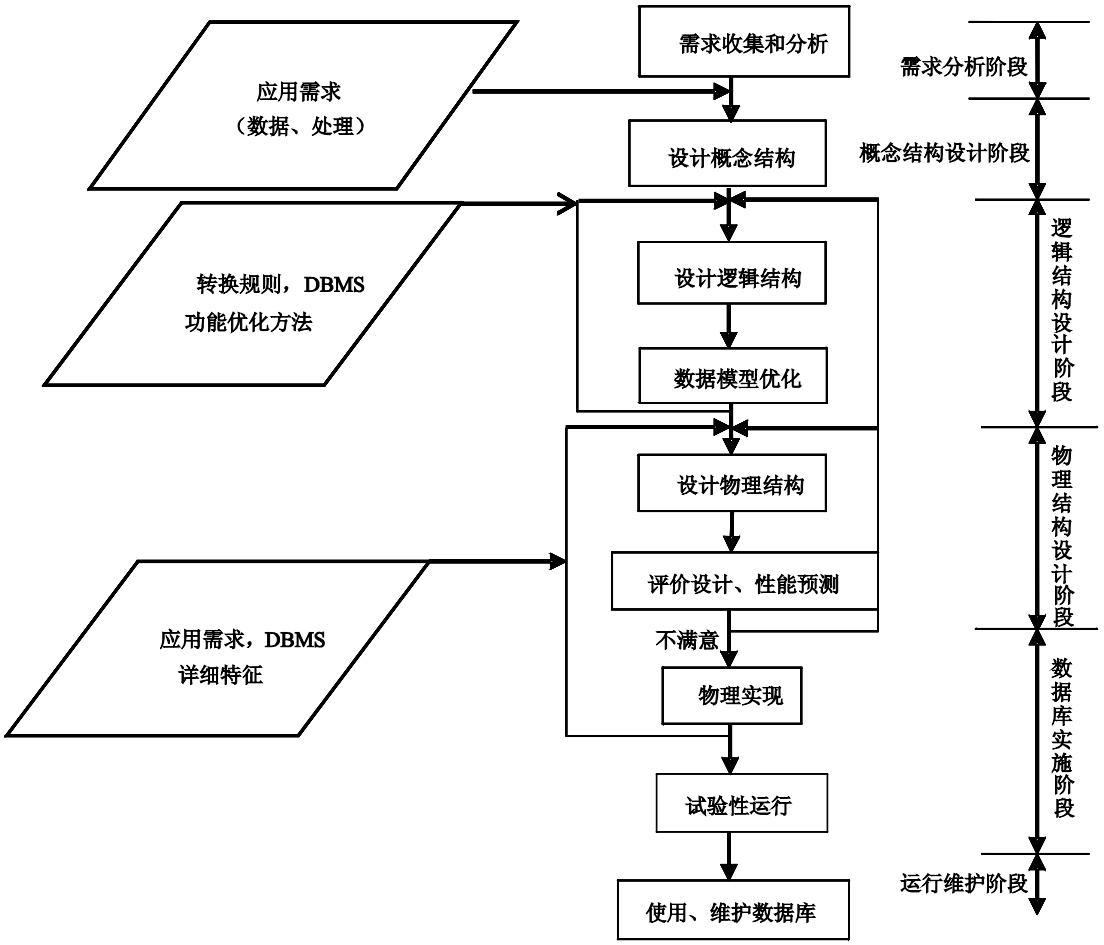


图 3-10　数据库设计的全过程

7. 数据库设计步骤

至今，数据库设计的很多工作仍需要人工来做，除了关系型数据库已有一套较完整的数据范式理论可用来部分地指导数据库设计之外，尚缺乏一套完善的数据库设计理论、方法和工具，以实现数据库设计的自动化或交互式的半自动化设计。所以数据库设计今后的研究发展方向是研究数据库设计理论，寻求能够更有效地表达语义关系的数据模型，为各阶段的设计提供自动或半自动的设计工具和集成化的开发环境，使数据库的设计更加工程化、更加规范化和更加方便易行，使得在数据库的设计中充分体现软件工程的先进思想和方法。

案例 2　学生成绩管理系统的登录

任务 1　设置系统编码格式

启动操作系统后运行 Eclipse，打开 Eclipse 编辑界面。为保证程序运行时不出现乱码情况，首先设置 Eclipse 的编码格式，保证多处编码格式统一，这里统一设置为 UTF-8 编

码格式，需要设置编码格式的几处地方为：

（1）窗口→首选项→常规→内容类型→Text→JSP 最下面缺省编码默认为“ISO-8859-1”，设置为 UTF-8 如图 3-11 所示。

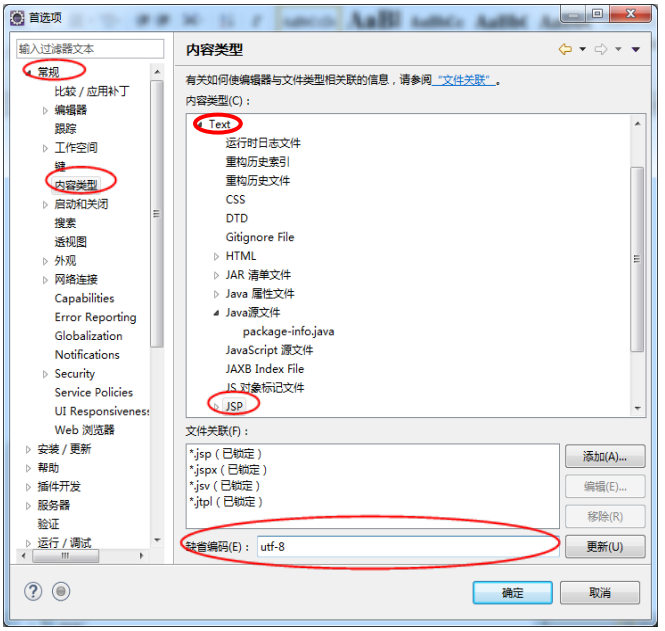


图 3-11　设置 JSP 文件的默认编码格式窗口

（2）窗口→首选项→常规→工作空间，工作空间的“文本文件编码”的缺省值为 GBK，单击“其他”按钮，选择“UTF-8”如图 3-12 所示。

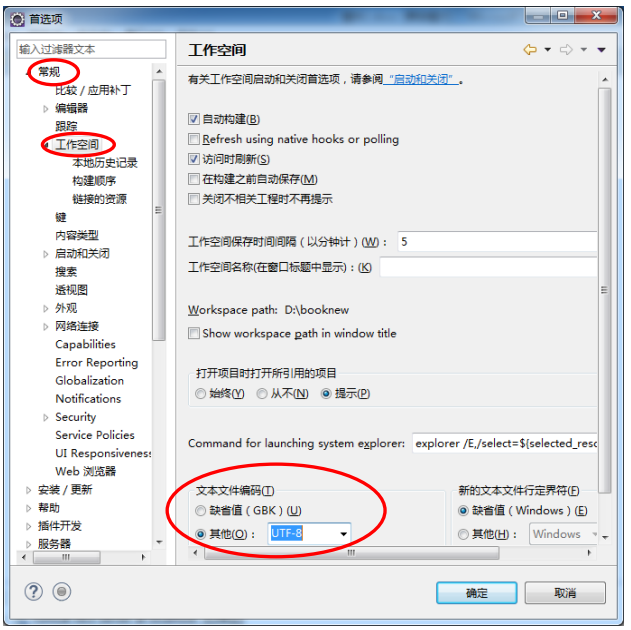


图 3-12　文本文件的默认编码格式窗口

（3） 窗口 → 首选项 → Web → JSP 文件面板，JSP 文件编码默认是 ISO-8859-1，单击编码右边的下拉式列表，选择 ISO 10646/Unicode(UTF-8)，如图 3-13 所示。

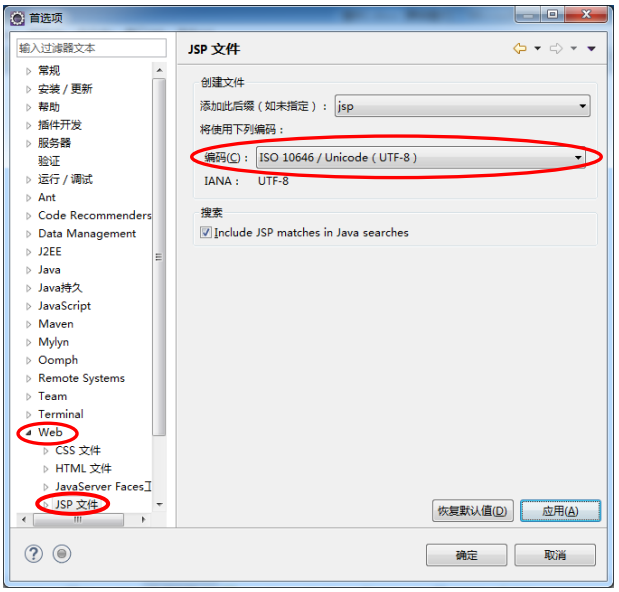


图 3-13　Web 中 JSP 文件的默认编码格式

任务 2　建立并运行Web项目

任务 2.1　建立动态 Web 项目

（1） 设置完编码格式，就可以开始建立 Web 项目了，在 Eclipse 主界面下单击“文件”菜单项选择“新建”操作项，在下拉菜单项中单击“动态 Web 项目”，如图 3-14 所示。

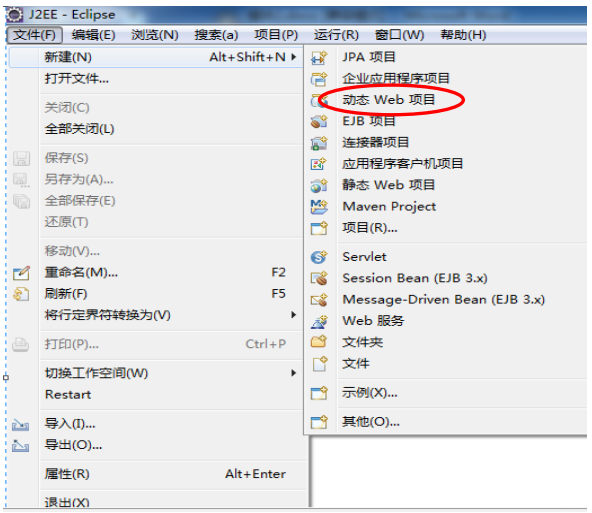


图 3-14　新建动态 Web 项目窗口

（2） 接下来在弹出新建 Web 项目对话框的项目名称（Project name）下，输入项目名称为 studentScorePro，把该项目存储位置设置为“E:\book\studentScorePro”，如图 3-15 所示。如果想更改项目的存储位置可以去掉 use default location 前的复选框，然后单击“location”按钮，在弹出的对话框中选择项目的存放路径。

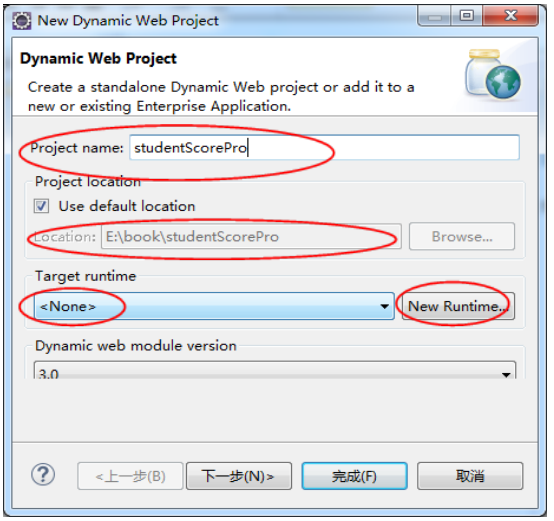


图 3-15　新建 Web 项目对话框

（3） 输入过项目名称后，在“Target runtime”选项中，单击右侧黑色箭头，若没有 Web 服务器配置，则显示为“<None>”，需要单击右侧“New Runtime”命令按钮，进行 Web 服务器配置，如图 3-16 所示。

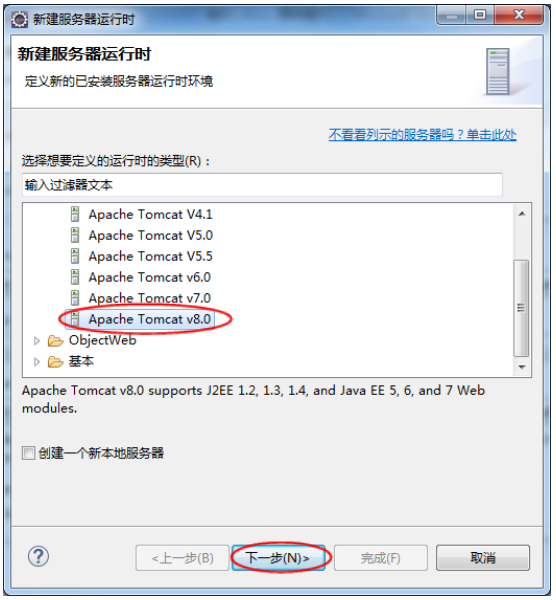


图 3-16　配置 Web 服务器

（4） 在弹出的窗口中选择安装的 Web 服务器“Apache Tomcat v8.0”版本，单击“下一步”按钮进入 Tomcat 环境配置，设置“Tomcat 安装路径”，单击“浏览”按钮，将安装路径设置为自己机器上 Tomcat 的安装路径，这里设置为 C:\Program Files\ Apache Software Foundation\Tomcat 8.0，Tomcat 环境配置对话框如图 3-17 所示。

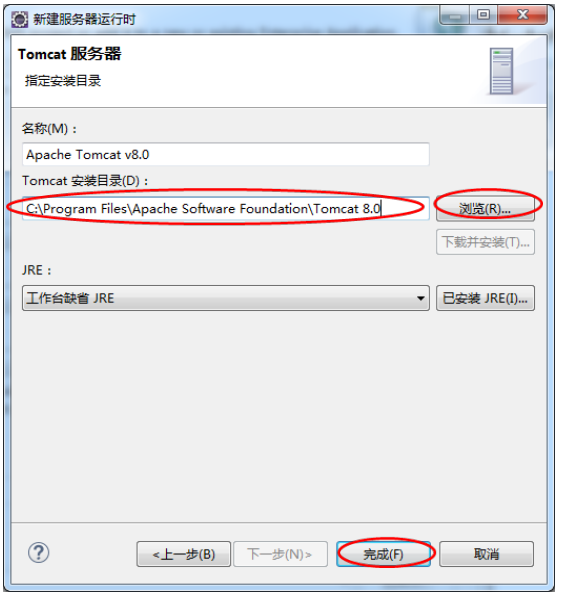


图 3-17　设置 Tomcat 安装路径

（5） 设置完安装路径后单击“完成”按钮回到图 3-15，设置完 Web 服务后的图 3-15 界面如下面图 3-18 所示。这时的“Target runtime”中已经出现了设置的 Web 服务器的信息。

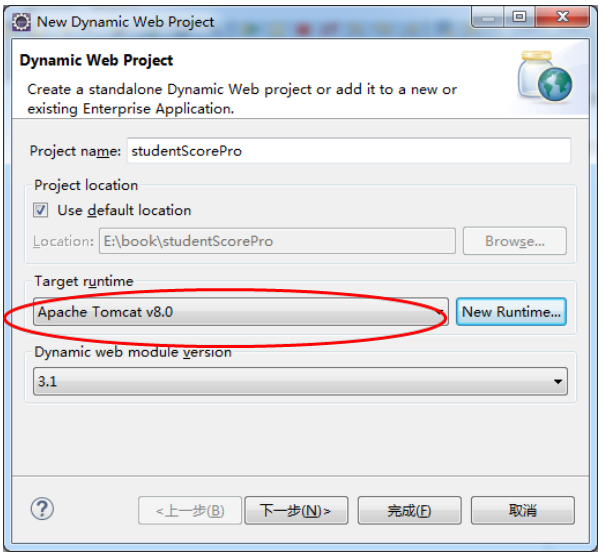


图 3-18　配置完 Web 服务器后的新建 Web 项目

（6） 然后再单击“下一步”按钮，进入图 3-19 设置“Source folders on build path”对话框，默认是“src”目录。图 3-19 中选项使用默认值，继续单击“下一步”按钮，进入图 3-20 配置 Web 模块设置窗口。

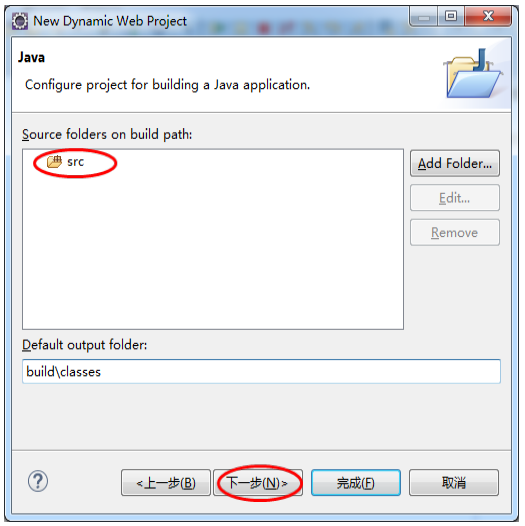


图 3-19　设置 Source folders on build path 对话框

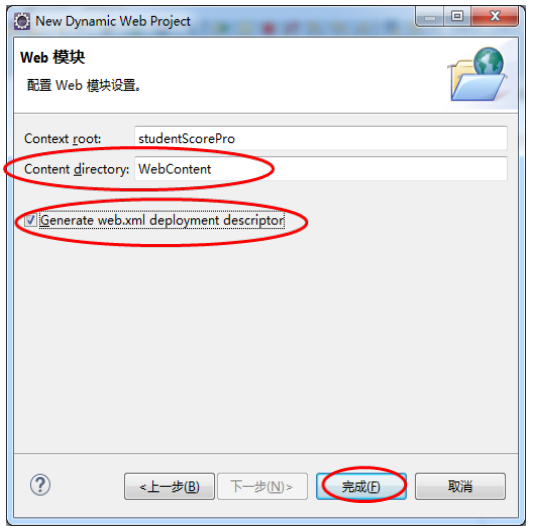


图 3-20　配置 Web 模块设置

（7） 在图 3-20 中，对于“Context root”和“Content directory”使用默认值即可，但是一定要将“Generate web.xml deployment descriptor”前的多选框选中，选中该选项的作用是创建 Web 项目时同时创建 Web.xml 文件，否则在创建 Web 项目时没有创建 Web.xml 文件，需要自己手动创建，单击“完成”按钮结束配置后，生成新的 Web 项目文件如图 3-21 所示。

图 3-21　查看创建的 Web 项目

任务 2.2　建立 login.jsp 登录页面

在项目资源管理器下选中任务 2.1 中建立的项目 studentScorePro 下的“WebContent”文件夹，单击鼠标右键弹出相应的快捷菜单如图 3-22 所示。

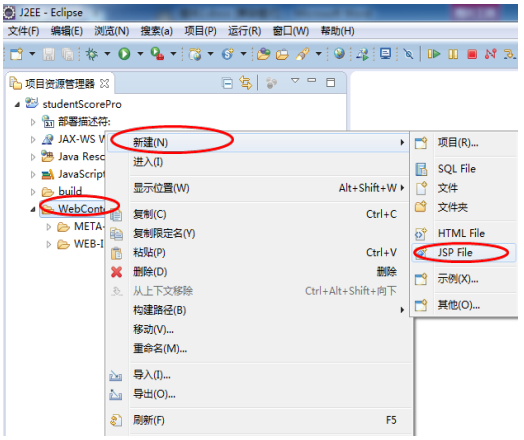


图 3-22　新建 JSP 文件窗口

在弹出的快捷菜单中选择“新建”菜单下的 JSP 文件类型“JSP File”，弹出新建 JSP 页面窗口如图 3-23 所示。

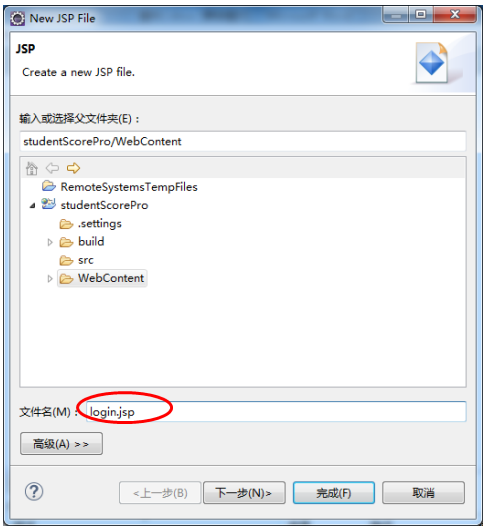


图 3-23　输入新建 JSP 文件名窗口

在输入新建 JSP 文件名窗口中，在“文件名”后的文本框中输入新建文件名为 login.jsp，单击“完成”按钮弹出代码编辑窗口如图 3-24 所示。

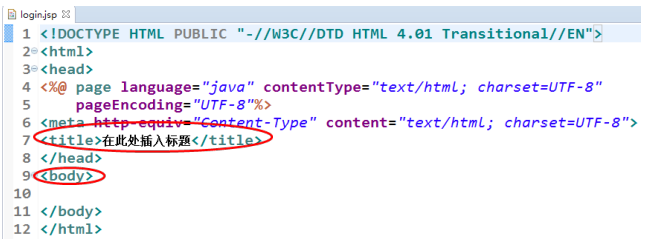


图 3-24　代码编辑窗口

在代码编辑窗口<title>标签中把其中的文字“在此处插入标题”修改为“登录页面”，在<body>标签后面，输入如下代码：

```
<H2 align="center">请登录</H2>
    <hr>
    <form   method="post" name="logform">
           <table border="1"   align="center">
                  <tr>
           <td align="center" style="font-size: 18px">用户账号：</td>
    <td><input type="text" name="userid" style="width: 160px" /></td>
                  </tr>
                  <tr>
<td align="center" style="font-size: 18px">密  码：</td>
    <td><input type="password" name="upass" style="width: 160px" /></td>
                  </tr>
                  <tr>
                         <td align="center">
                         <input type="submit" value="登录"
                                style='font-size: 18px'  /></td>
           <td align="center"><input type="reset" style='font-size: 18px'
                                value="清空"></td>
                  </tr>
           </table>
    </form>
```

任务 2.3　修改 Web.xml 文件

代码输入完毕后，在对象资源管理器中找到“WebContent”文件夹下的“WEB-INF”文件夹并展开，双击其中的“web.xml”文件，则进入 web.xml 文件的编辑界面，如图 3-25 所示。

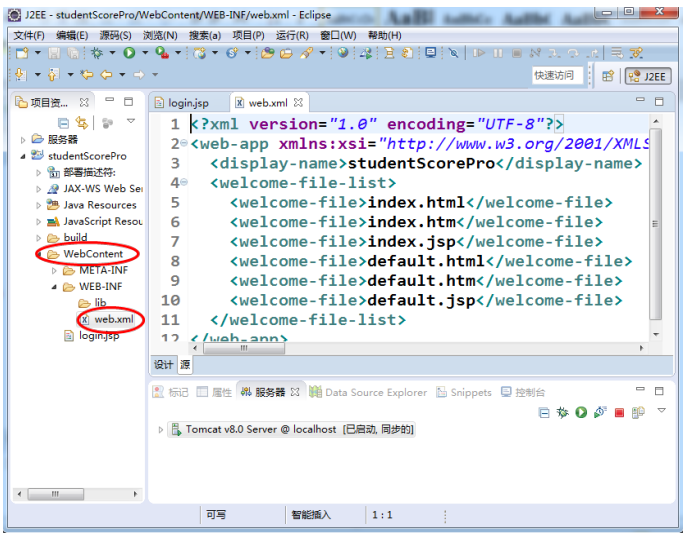


图 3-25　web.xml 文件编辑窗口

如果打开的 web.xml 文件不是如图 3-25 的编辑界面，而是打开如图 3-26 的界面，单击其中的“源”标签切换到如图 3-25 的编辑窗口。

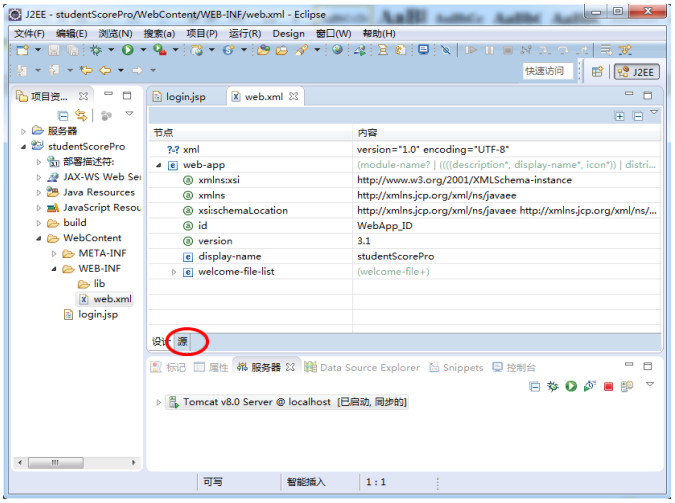


图 3-26　web.xml 文件设计窗口

web.xml 是网络程序中的一个很重要的配置文件。它是用来初始化配置信息：比如 Welcome 页面、servlet、servlet-mapping、filter、listener 启动加载级别等。每个 xml 文件都有定义它书写规则的 Schema 文件，也就是说 JavaEE 定义的 web.xml 所对应的 xml Schema 文件中，定义了多种标签元素，但有些不是很常用的，只需记住一些常用的并知道怎么配置就可以了。web.xml 文件的其他功能可以查阅相关资料，这里先介绍其指定欢迎界面的功能。

例如：

```
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
        <welcome-file>welcome.jsp</welcome-file>
</welcome-file-list>
```

上面的 web.xml 指定了 2 个欢迎页面，显示时按顺序从第一个找起，如果第一个存在，就显示第一个，后面的不起作用。如果第一个不存在，就找第二个，以此类推。如果指定的所有欢迎页面文件都不存在的话，运行程序会出现 404 错误，如图 3-27 所示。

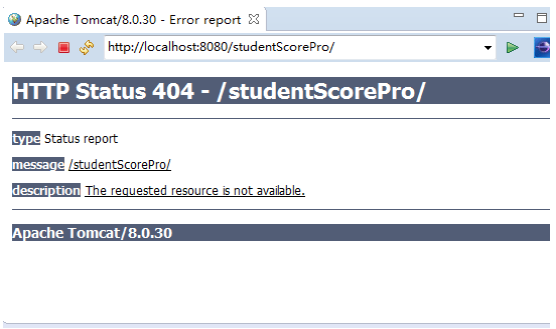


图 3-27　运行找不到文件错误窗口

关于欢迎页面，访问一个网站时，默认看到的第一个页面是欢迎页面，一般情况下是由首页来充当欢迎页面的，通常会在 web.xml 中指定欢迎页面，所以，默认创建的动态 Web 工程在 WEB-INF 文件夹下面都有一个 web.xml 文件，如果没有需要手动建立，这在任务 2.1 建立 Web 项目中已经介绍。web.xml 文件的内容要根据自己建立的系统情况进行修改，这里以上面建立的 login.jsp 为欢迎界面，所以图 3-25 其中的一个<welcome-file>标签的值为 login.jsp，如图 3-28 所示。

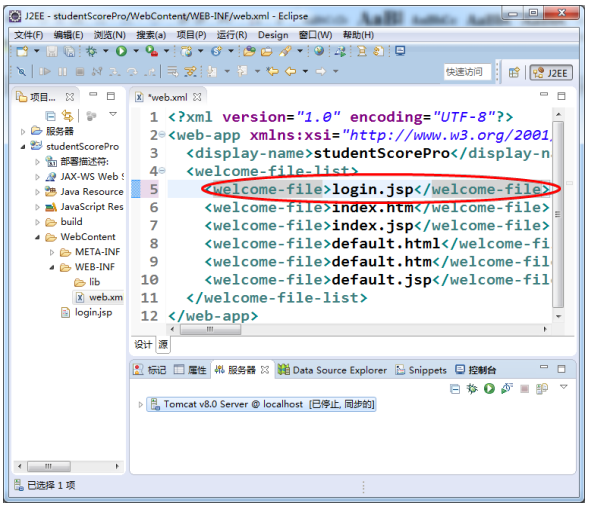


图 3-28　修改后的 web.xml 文件

任务 2.4　运行 Web 项目

配置完 web.xml 文件后，下面就可以运行上面建立的项目了。在项目资源管理器下，选中所建项目 studentScorePro 并单击右键，弹出该项目的快捷菜单如图 3-29 所示。

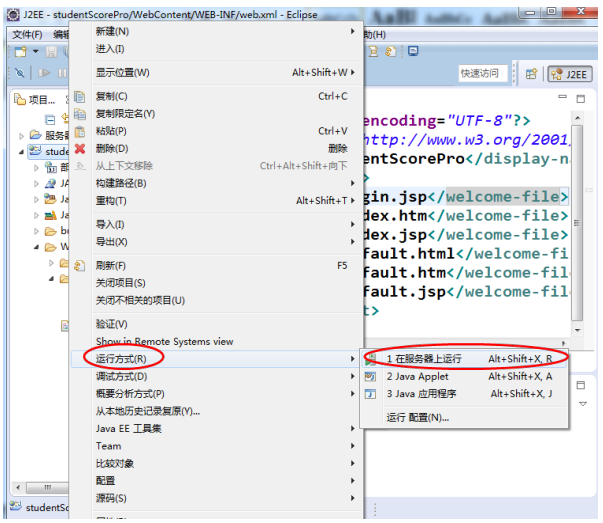


图 3-29　运行 Web 项目

在快捷菜单中，单击“运行方式”菜单下的“在服务器上运行”项，继续弹出服务器运行对话框如图 3-30 所示。

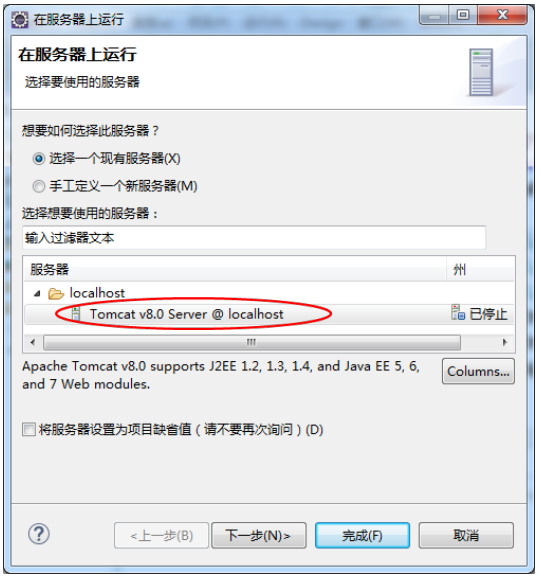


图 3-30　在服务器上运行对话框

在“服务器上运行”对话框中，选择已经安装的 Tomcat 服务器“Tomcat v8.0 Server at localhost”后，单击“完成”按钮运行程序，出现登录界面如图 3-31 所示。

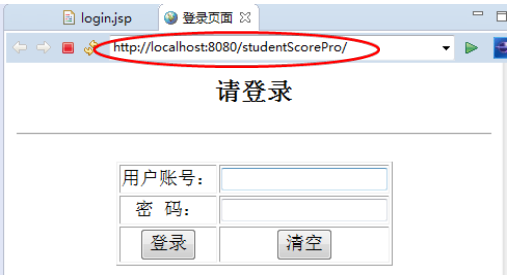


图 3-31　用户登录界面

在用户登录界面中的地址框中，地址 http://localhost:8080/studentScorePro/，该地址中各项的含义是：“localhost”代表本机，也可以写成 IP 地址“127.0.0.1”，“8080”代表 Tomcat 服务器端口号，“studentScorePro”代表运行的项目名称。

在登录过程中为了避免用户没有输入用户账号和密码进行登录，所以在用户单击登录按钮时，要进行用户账号或密码是否为空的验证。具体代码是在 login.jsp 页面代码中的</head>标签后<body>标签前输入如下代码：

```
<script type="text/Javascript">//Javascript 代码
   function checksubmit() {//定义函数 checksubmit()
          with (document.logform) {
                 var uid = userid.value;//读取用户输入的账号
```

```
                    var pwd = upass.value;//读取用户输入的密码
                    if (uid == null || uid == "") {//如果没有输入账号，或账号为空
串
                        alert("请填写登录账号 !");//弹出请填写登录账号!消息框
                        return;
            } else if (pwd == null || pwd == "") {//如果没有输入密码，或密码为
空串
                        alert("请填写密码!");//弹出请填写密码!消息框
                        return;
                    } else
                        document.logform.submit();//提交窗体
            }
        }
    </script>
```

同时在 login.jsp 中找到如下代码：

```
    <input type="submit" value="登录"
                                style='font-size: 18px'  /> 并在“ /> ”符号前
加入如下代码，onClick="checksubmit()"
```

运行项目实现如图 3-32 所示效果。

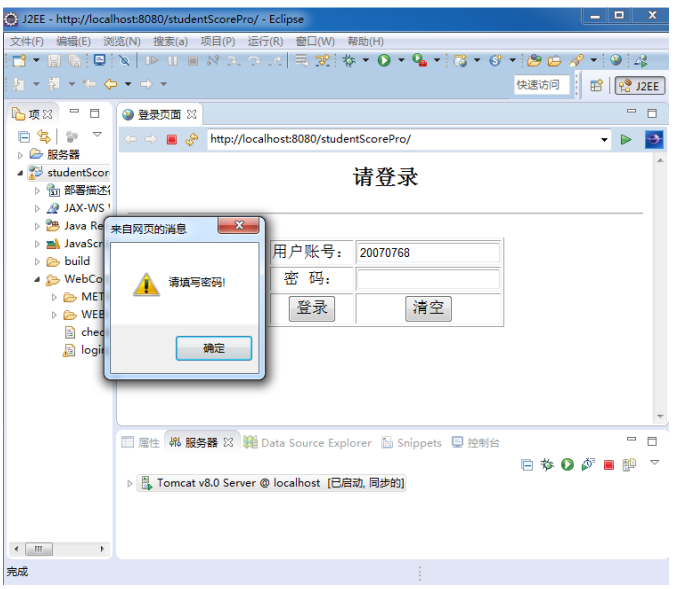


图 3-32　用户登录密码为空消息提示窗口

任务 2.5　建立 checklogin.jsp 登录验证界面

在“登录界面代码”的“form”标签内，加入“action”属性：

```
    <form action="checklogin.jsp"  method="post" name="logform">
```

然后，在项目资源管理器中，按照创建“login.jsp”文件的步骤，创建“checklogin.jsp”

文件，打开checklogin.jsp文件的编辑窗口，在<html>标签上面加入导入连接数据库对象所在的包，具体代码是：

```
<%@page import="Java.sql.ResultSet"%>
<%@page import="Java.sql.Statement"%>
<%@page import="Java.sql.Connection"%>
<%@page import="Java.sql.DriverManager"%>
```

把其中的<title>标签中“在此处插入标题”，修改为“登录验证”，并在<body>标签后输入如下代码：

```
<%String userid = request.getParameter("userid").trim();
//读取登录界面 login.jsp 中输入的用户账号
String upass = request.getParameter("upass").trim();
//读取登录界面 login.jsp 中输入的密码
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");// 加 载 SQL Server 2008 驱动
Connection conn = DriverManager.getConnection(
"jdbc:sqlserver://localhost:1433;DatabaseName=ssms ","sa","hh123456");
/*创建连接对象，localhost 代表本级也可以用 ip 地址 127.0.0.1 代替，1433 是 SQL Server 端口号，databaseName 用于指定访问的数据库名，sa 是 SQL Server 登录账号，hh123456 是 sa 用户的登录密码 */
Statement stmt = conn.createStatement();//创建语句对象 stmt
String sql = "select * from teacher where tid= '"+userid+"' and tPassword="+upass;
//定义查询语句 sql 从教师表中读取用户编号等于输入编号、密码等于输入密码的教师信息
ResultSet rs = stmt.executeQuery(sql);
//语句对象 stmt 执行查询语句 sql，将返回结果存放在数据集 rs 中
if (rs.next() == true){//验证用户在登录界面中输入的账号和密码与后台数据库中存储的账号和密码是否一致
session.setAttribute("uid", userid);//把用户账号存入 session，以备后面页面读取
session.setAttribute("upass", upass);//把用户密码存入 session，以备后面页面读取
out.print("<script> alert('登录成功！') </script>");//弹出登录成功提示框
}
else
{//弹出错误 窗口，页面重新跳转到登录页面
out.print("<script>alert('用户名或密码错误！请重新输入'); location.href='login.jsp'</script>");
}
rs.close();//关闭数据集
stmt.close();//关闭语句对象
conn.close();//关闭连接对象
//注意：关闭这三个对象顺序不要颠倒
%>
```

任务 2.6　导入 JDBC 驱动包

JSP 连接 SQL Server 2008 数据库必须加载 JDBC 驱动包才能正确运行，所以在运行访问 SQL Server 2008 数据库程序之前，先加载 JDBC 驱动包。可以从百度中搜索 sqljdbc 进

行下载，或通过如下链接进行下载 https://yunpan.cn/cPL8iTWpNVcrI，密码是 774c。sqljdbc 文件有两个分别是：sqljdbc.jar 和 sqljdbc4.jar ，两者的区别是，sqljdbc.jar 类库要求应用 5.0 版的 Java 运行时环境 (JRE)。连接到数据库时，在 JRE 6.0 上应用 sqljdbc.jar 会引发异常。sqljdbc4.jar 类库要求应用 6.0 或更高版本的 Java 运行时环境 (JRE)。在 JRE 1.4 或 5.0 上应用 sqljdbc4.jar 会引发异常。要根据自己机器安装的 JDK 版本信息下载相应的驱动包，这里用的 JDK 版本是 1.8 所以选择 sqljdbc4.jar。下载完 sqljdbc4.jar，必须加载该驱动包，具体操作过程如下：

（1） 在建立的项目上单击鼠标右键，在弹出的快捷菜单中选择“构建路径”，然后选择“配置构建路径”，如图 3-33 所示。

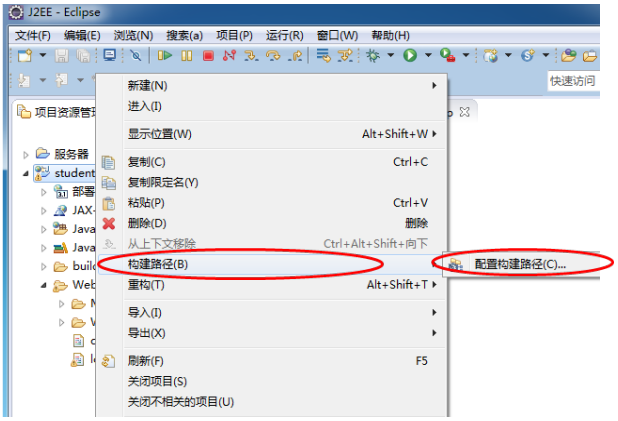


图 3-33　选择构建路径窗口

（2） 选择了配置构建路径会弹出如图 3-34 所示的“配置 Java 构建路径”窗口，在该窗口中选择“库”标签，在单击右边的“添加外部 JAR(X)”命令按钮，会弹出如图 3-35 所示的“查找 JAR 文件”对话框。找到下载的 sqljdbc4.jar 文件，单击“打开”按钮，又回到图 3-34 界面，选择的 jar 包文件已经添加进来，如图 3-36 所示的导入 sqljdbc4.jar 包后构建路径对话框，单击“确定”按钮，完成 JDBC 驱动包的导入。

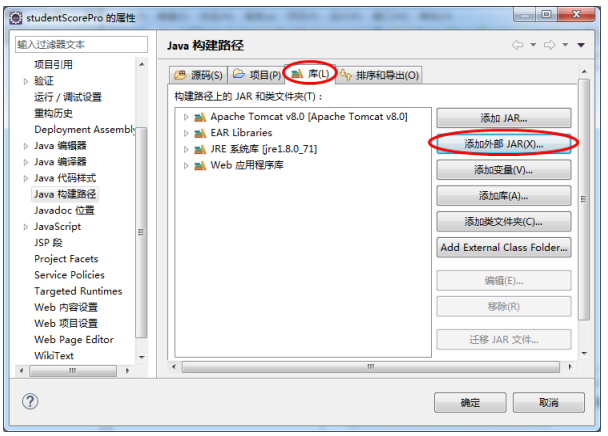


图 3-34　配置 Java 构建路径窗口

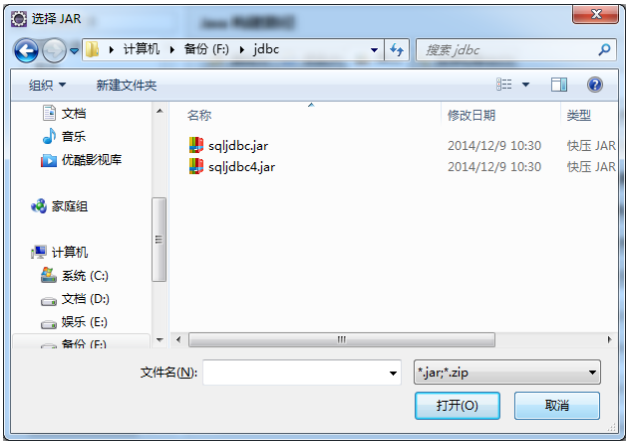


图 3-35　选择 JAR 文件对话框

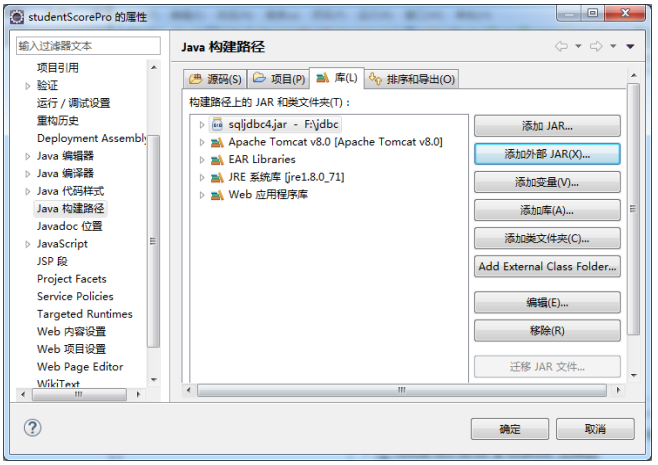


图 3-36　导入 sqljdbc4.jar 包后构建路径对话框

导入 jdbc 驱动包后，就可以运行项目，访问后台数据库了，运行项目输入正确密码登录后运行效果如图 3-37 所示，输入了错误密码运行效果如图 3-38 所示。如果在导入 sqljdbc4.jar 包后，程序运行仍然找不到 JDBC 驱动，则需要把 sqljdbc4.jar 包复制到 Tomcat 安装路径的 lib 目录下，例如根据本机 Tomcat 安装情况，sqljdbc4.jar 包复制到的具体路径是 C:\Program Files\Apache Software Foundation\Tomcat 8.0\lib。

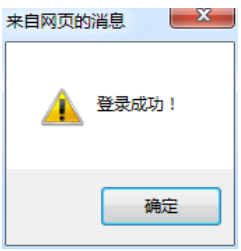


图 3-37　登录成功提示

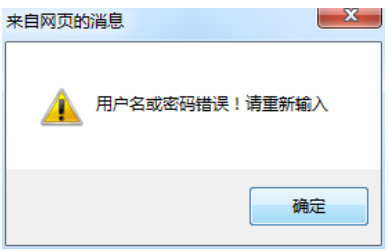


图 3-38　用户名或密码输入错误消息提示窗口

习题

认真思考案例 2 的设计与实现过程，设计并实现学生用户登录功能。

相关知识

一、select 语句

数据查询是数据库的核心操作。SQL 提供了 select 语句进行数据查询，该语句的一般格式是：

```
SELECT [ALL|DISTINCT]  [TOP N]  [PERCENT] <目标列表达式> [,<目标列表达式>] …
FROM <表名或视图名>  [,<表名或视图名>]  [(SELECT 语句> [[AS] <别名>])]
[WHERE <条件表达式>]
[ORDER BY <列名 1> [ASC|DESC]]
[GROUP BY <列名 2> [HAVING <条件表达式> ]
```

说明：方括号“[]”表示该项为可选项，在应用中根据需要可有可无；尖括号“<>”表示该项为必选项，只要用到查询语句，该项必须有；竖线“|”表示二选一，在应用中根据需要选择竖线左边或者竖线右边的项。

select 语句的含义是根据 where 子句的条件表达式从 from 子句指定的基本表或者视图或者派生表中查找满足条件的记录，再按 select 子句的目标列表达式选出所有记录或者去除重复的记录或者记录的前 n 个或者前 n%条记录。如果有 order by 子句，则将查询结果按<列名 1>升序或者降序进行排序。如果有 group by 子句，则将查询结果按<列名 2>的值进行分组统计，而 having 是指定分组统计时的过滤条件。

应用 1　目标列的变化

【例 1】查询指定列。

从学生表 student 中查询学生的学号及姓名。

```
select sid,sName
from student ;
```

在查询中，如果需要将所查表中指定列给出，则直接给出相关列名即可。

【例 2】查询所有列。

从学生表 student 中查询所有学生的信息。

```
select *
from student ;
```

或者

```
select sid,sPassword,sName,sClassName,sSex,sBirthday,sPolitical,
fresh,sPhotoPath,sEmail
from student
```

在查询中，如果需要将所查表中全部列给出，并且列的次序按表中次序，同时列也没有重命名的要求，则可不必将列名一一写出，直接采用“*”即可。

【例 3】对列重命名。

从学生表 student 中查询学生的学号及姓名，同时，将列名以汉字的形式给出。

```
select sid  as 学号,sName as 姓名
from student ;
```

或者

```
select sid  as  "学号",sName as  "姓名"
from student ;
```

或者

```
select sid  as  '学号',sName as  '姓名'
from student
```

或者

```
select  sid  "学号",sName  "姓名"
from student ;
```

或者

```
select sid  学号,sName 姓名
from student ;
```

在应用系统的数据库实现时，一般列名的命名采用英文表示，但用户在使用应用系统时，通常不明白众多英文表示的含义，因此，通常可以采用以上四种方式之一将查询的结果给出中文列名。

【例 4】对列进行计算。

从学生表 student 中查询学生的姓名及年龄。

```
select sName,year(getdate())-year(sBirthday) sSage
from student ;
```

在数据库存储时，为了节省存储空间，可以省去一部分数据的存储，例如学生的出生年份，在学生表 student 中，仅存放学生的年龄，当需要学生的出生年份信息时，可以采用对列计算的方式得出。

【例 5】去重复。

从学生表 student 中查询学生的性别，以确保输入数据的正确性。

```
select distinct(sSex)
from student ;
```

在数据查询过程中，所查表中不存在完全相同的两行，但因为查询语句所指定的列导致查询的结果可能会出现完全重复的两行或者多行，在这种情况下，采用 distinct 短语去除查询结果中重复的行。

【例 6】取查询结果的前 n 行。

从学生表 student 中查询排在前 10 个的学生信息。

```
select top(10)*
from student ;
```

在数据查询过程中，查询结果可能非常多，但对于实际应用来说，可能仅需要个别的一些数据行，例如任课教师查询本次考试成绩的前三名，针对这种需要，top 短语实现输出查询结果的前 n 行数据。

【例 7】取查询结果的前 n%行。

从学生表 student 中查询前 10%个学生的信息。

```
select top 10 percent *
from student ;
```

类似于【例 6】，在查询结果中仅需要结果的百分之多少数据行，此时，可以使用 top 短语与 percent 短语结合的方式实现。

【例 8】聚集函数的应用。

统计学生表 student 中学生人数。

```
select COUNT(sid)
from student;
```

count 函数的作用是统计行数或者某一列中值的个数。

统计成绩表 score 中最高分。

```
select MAX(totalScore) 最高分
from score;
```

max 函数的作用是求一列值中的最大值，与其对应的是 min 函数，min 函数的作用是求一列值中的最小值。

统计所有课程的总分。

```
select SUM(totalScore) 总分
from score;
```

sum 函数的作用是求一列值的总和，但此列的数据类型必须是数值类型。

统计所有课程的平均分。

```
select avg(totalScore) 平均分
from score;
```

avg 函数的作用是求一列值的平均值，同样此列的数据类型必须是数值类型。

应用 2　条件表达式的变化

【例 1】关系表达式构成的条件。

从学生表 student 中查询所有女生的信息。

```
select *
from student
where sSex='女' ;
```

关系运算符有＝（等于）、>（大于）、<（小于）、>=（大于等于）、<=（小于等于）、!=或者<>（不等于）、!>（不大于）、!<（不小于）八个，由它们构成的表达式作为 select 语句的查询条件，例如 sAge>18、sAge!=18 均是合法的查询条件。

【例 2】逻辑表达式构成的条件。

从成绩表 score 中查询课程编号是“3341103314”并且期末总成绩达到及格以上要求的成绩信息。

```
select *
from score
where cid='3341103314' and totalScore>=60 ;
```

逻辑运算符有 not（逻辑非）、and（逻辑与）、or（逻辑或）三个，由它们构成的表达式也是 select 语句的查询条件，一般称为复合查询条件，例如 sAge>18 and sSex='女'、sAge>18 or sSex!='女'是合法的查询条件。

【例 3】区间短语条件。

从成绩表 score 中查询课程编号是“3341103314”并且期末总成绩在[90，100]的成绩信息。

```
select *
from score
where cid='3341103314'  and totalScore between 90 and 100 ;
```

与该语句等价的是：

```
select *
from score
where cid='3341103314' and totalScore>=90 and totalScore<=100 ;
```

短语 between…and…和 not between…and…的作用是查询列值在（或者不在）指定闭区间（即[n1,n2]）内的数据，其中 between 后是区间的下限，and 后是区间的上限，且勿颠倒。对 between…and…和 not between…and…这两个短语可以由逻辑运算符 and 构成的逻辑表达式替换，结果相同。

【例 4】集合短语条件。

从成绩表 score 中查询学号是“1234120101”或者学号是“1234120103”或者学号是“1234120107”三个学生的成绩信息。

```
select *
from  score
where sid in ('1234120101','1234120103','1234120107') ;
```

与该语句等价的是：

```
select *
from score
where sid='1234120101' or sid='1234120103' or sid='1234120107';
```

短语 in 的作用是查询列值在指定集合内的数据，与其对应的是 not in 短语，not in 短语的作用是查询列值不在指定集合内的数据。同样，这两个短语也可以由逻辑运算符 or 构成的逻辑表达式替换，查询结果相同。

【例 5】字符匹配条件（又称模糊条件）。

从学生信息表 student 中查询姓刘的所有学生信息。

```
select *
from student
where sName like '刘%';
```

从学生信息表 student 中查询姓名是两个字的所有学生信息。

```
select *
from student
where sName like '__';
```

从学生信息表 student 中查询姓名的第二个字是“明”字的所有学生信息。

```
select *
from student
where sName like '_明%';
```

在实际应用中，通常用户是十分清楚要查询的数据是什么，因此，模糊查询是常用的查询方法，即使用 like 短语查询指定的列值与“匹配串”相匹配的数据行。短语 like 的一般使用格式如下：

```
列名 Like '匹配串'[escape '换码字符']
与 like 短语对应的短语是 not like，其一般使用模式如下：
列名 not Like '匹配串'[escape '换码字符']
```

“匹配串”可以是一个确定的字符串，也可以是包含通配符%（百分号）和_（下列线）的字符串。通配符%百分号和_（下划线）的含义是：

%（百分号）：代表任意长度的字符串，当然也可以是长度为 0 的字符串。例如：

sName like '刘%冰'表示学生姓名中，姓“刘”的，并且名字的最后一个字是“冰”的学生，这其中就包括有名字是两个字的“刘冰”、名字是三个字或者三个字以上的，如“刘冰冰”、“刘杨冰冰”等都是满足该查询条件的姓名。

_（下划线）：代表任意一个字符。例如：

sName like '刘_冰'表示学生姓名中，姓“刘”的，姓名由三个字构成的学生。如“刘冰冰”、“刘爱冰”等是满足该查询条件的姓名，但“刘冰”是不满足查询条件的姓名。

sName not like '刘%'表示学生姓名中不姓“刘”的学生。

如果在查询的字符串本身中含有通配符“%”或“_”，则必须使用 escape '换码字符'对通配符进行转义。例如：

查询课程名为“高等数学_1”的课程信息。

```
select *
from course
where cName like '高等数学\_1' escape '\';
```

在该查询语句中，查询字符串后使用 escape '\'约定跟在“\”后的字符不是通配符，而是普通字符“_”。换码字符没有明确规定用哪个字符，但通常使用与所查字符不同的字符即可。

```
select *
from course
where cName like '高等数学 a_1' escape 'a';
```

在该查询语句中，换码字符使用的是“a”。

```
select *
from course
where cName like '高等数学__1' escape '_';
```

在该查询语句中，换码字符使用的是“_”，第一个“_”代表换码字符，第二个“_”是所查字符。

因此，上面三个查询语句执行的结果是相同的。

注意：只有紧跟在换码符后面的字符才被转义为普通字符。

【例 6】空值条件。

查询期末成绩为空的成绩信息。

```
select *
from score
where totalScore is NULL;
```

空值是“不存在”、“不知道”或“无意义”的值，但绝对不能说空值是零。例如，某学生，平时课程学习非常好，但在期末考试时因为个人原因不能在校参加考试，这时，该生的期末成绩是不存在的，即为空，而不是零。查询空值使用短语 is 进行。

与其对应的是 is not，即非空值。

```
select *
from score
where totalScore is not NULL;
```

注意：空值的查询必须使用“is”，绝对不能采用“=”替换。

应用 3　查询结果次序的变化

【例 1】从学生信息表 student 中按照年龄升序查询所有信息。

```
select *
from student
order by year(getdate())-YEAR(sBirthday);
```

或者

```
select *
from student
order by year(getdate())-YEAR(sBirthday) asc;
```

order by 子句的作用是将查询结果按照一列或者多列进行升序或降序排序。升序采用关键字 asc，降序采用关键字 desc，若不写任何关键字，则采用默认次序——升序。

【例 2】从学生信息表 student 中按照姓名降序、学号升序查询所有信息。

```
select *
from student
order by sName desc,sid asc;
```

或者

```
select *
from student
order by sName desc ,sid;
```

此例中，先按照姓名降序进行排序，若出现同名现象，则对同名学生按照学号升序排列。

注意：order by 子句中列名与次序关键字（asc 或者 desc）是一对一的关系，若出现如

```
select *
from student
order by sName ,sid desc;
```

的情况，则表示学生姓名 sName 列按照升序排列，而不是先按照 sName 降序排列，若相同再按学号 sid 降序排列。

应用 4　查询结果分组信息处理

【例 1】查询每个学生选修课程的门数。

```
select sid,COUNT(*)
from score
group by sid;
```

group by 子句的作用是将查询结果这个大的集合再进一步细分成若干个小集合，然后再对每个小集合进行数据处理。在本例中，首先从成绩表 score 中取出所有人的所有成绩，形成一个大的集合，按照学号对这个大集合再细分为若干个小集合，即每个小集合是一个学生所选修的课程形成的，然后对每个小集合采用 count 函数统计其个数，最终得出每个学生选修课程的门数。

【例 2】查询选修人数超过 30 人的课程编号及选课人数。

```
select cid,COUNT(*) 总人数
from score
group by cid having COUNT(*)>=30;
```

在 group by 子句的后面，带上 having 短语后表明了选择小集合的条件，having 短语后

面一般会给出条件。如在本例中，只有小集合中记录个数大于等于 30 时，该小集合才能最终给出。

【例 3】查询每个学生选修课程得分在 90 分以上的课程门数。

```
select sid,COUNT(*) 成绩 90 分以上门数
from score
where totalScore>=90
group by sid ;
```

注意：

having 与 where 相似之处均用于设定查询条件。

having 与 where 的区别在于二者作用对象和执行的优先级别不同。

where 作用于查询的基本表、视图或者派生表，而 having 作用于查询出来的数据集合。

在语句 select 的各子句执行的优先级别上，where 优先于 having 先执行。例如：

```
select sid,COUNT(*) 有两门以上课程成绩在 90 分以上
from score
where totalScore>=90
group by sid having count(sid)>=2;
```

若成绩表 score 没有建立任何索引，在执行该语句时，先对 score 表中的数据一行一行地进行判断是否满足条件“totalScore>=90”，如果满足该条件，则将该数据行取出；当对成绩表 score 扫描一遍后，得到的数据集全部满足条件“totalScore>=90”，接下来就对该数据集按学号 sid 进行分组，得到一个个的小集合，然后再对每个小集合判断是否满足条件“count(sid)>=2”，即是否有两门课程以上是 90 分以上的成绩，如果是，则将该小集合返回，否则，舍弃。

应用 5　表表达式的变化

【例 1】查询学生表 student 中性别是“女”并且姓“刘”的所有学生的学号及姓名。

```
select sid,sName
from student
where sSex='女' and sName like '刘%';
```

在本例中，查询目标是从学生表 student 中进行的，这种查询一般称为单表查询。

【例 2】查询学生的姓名、选修的课程名及学生每门课程的成绩。

```
select sName,cName,totalScore
from student,score,course
where student.sid=score.sid and course.cid=score.cid;
```

在本例中，查询目标是从学生表 student、课程表 course 和成绩表 score 三张表中进行的查询，这种查询一般称为多表查询或者连接查询，多表查询也是应用中较常用的。当多表查询中条件运算符用的是等号“＝”时，称为等值连接查询，如果用的不是等号“＝”，则称为非等值连接查询，一般在应用中多数是等值连接查询。

【例 3】查询院系视图 sdept 的所有信息。

```
select *
from sdept;
```

采用 select 语句对视图的查询格式与对基本表的查询相同。视图将在本案例的拓展知识中学习到。

【例 4】查询信息工程学院男生与女生的平均年龄。

```
select sSex,avg(sAge) 平均年龄
from
(select sName,sSex,year(getdate())-year(sBirthday) sAge
from student,department
where dName='信息工程学院' and did=substring(sid,3,2) )
as Information
group by sSex;
```

或

```
select sSex,avg(sAge) 平均年龄
from
(select sName,sSex, year(getdate())-year(sBirthday) sAge
from student,department
where dName='信息工程学院' and did= substring(sid,3,2) )
group by sSex;
```

在本例中，from 的后面不是一个基本表或者视图，而是另外一个查询语句，这种查询称为是对内联表查询。内联表查询也称为派生表查询或者子表查询，是在外部查询的 from 子句中定义的。内联表的存在范围为定义它的外部查询，只要外部查询一结束，内联表就不存在了。定义内联表的查询语句要写在一对圆括号内，可以为内联表命名，也可以不再给内联表命名。本例中语句的执行过程是：先对 from 后面的查询语句执行，得到的查询结果作为内联表，其名字为 Information，接下来对内联表 Information 按照性别 sSex 执行分组，最后对每个分组读取性别 sSex，同时使用函数 avg 求年龄 sAge 的平均值。

【例 5】先建立表值函数 FindScore，通过表值函数 FindScore 查询考试成绩是 90 分的信息。

```
create function FindScore
(
@grade int
)
returns table
as
return
 select * from score where totalScore=@grade
go
select * from FindScore(90);
```

create function 是创建函数 FindScore 的关键字，关于自定义函数知识，参阅相关资料。

内联表值函数是一种可重用的表表达式，能够支持输入参数。除了内联表值函数支持输入参数以外，内联表值函数在其他方面都与视图相似。

应用 6　子查询

【例 1】查询学生姓名为“刘冰冰”（假设姓名没有与其重复的学生）的学号和课程的平均成绩。

```
select sid,avg(totalScore) 平均成绩
from score
where sid=(select sid from student where sName='刘冰冰');
```

按照子查询返回的结果，将子查询分为三类：单行子查询、多行子查询和多列子查询。使用子查询的原则：

（1） 子查询放在圆括号中；

（2） 将子查询放在比较条件的右边，以增加可读性。

（3） 在子查询中，可以使用两种比较条件：单行运算符和多行运算符。

如果子查询能够保证返回值是 1 行或者 0 行，则可采用单行比较符：=、<、>、<=、>=等；否则要采用多行比较符：in、>any 、>all、<any、<all 等。

注意，在一个查询语句中，最多只能有一个 order by 子句，order by 通常写在最外层查询中，子查询不能带 order by 子句。

带有子查询的查询语句又称为嵌套查询。

在本例中，使用的是单行子查询。单行子查询指从子查询或（内查询）中返回是一行的查询。本例的查询语句由两部分构成：子查询（内查询）和外查询。在执行过程中，先执行子查询，查询结束后，将学生“刘冰冰”的学号“1234120101”返回给外查询，然后外查询从成绩表 score 查询出学号为 1234120101 的所有成绩信息，最后返回其学号以及所选课程的平均成绩。

【例 2】查询院系名称含有“工”字院系的学生姓名、学生年龄和学生班级。

```
select sName,year(getdate())-year(sBirthday) sAge,sClassName
from student
where substring(sid,3,2) in (select did from department
where dName like '%工%');
```

在本例属于多行子查询。多行子查询指子查询返回结果是多行的查询。本例的查询语句同样由子查询和外查询两部分构成，在语句的执行过程中，先执行子查询从院系表 department 查找所有院系名称中带有“工”字的院系，例如信息工程学院、建筑工程学院、电子工程学院等；将查询结果中的院系编号集合返回给外查询；然后外查询从学生表中逐一取出每个学生的信息后，采用取子串函数截取学号中的第 3 位和第 4 位，即院系编号，判断该院系编号是否在子查询的结果集中，如果在，则返回该生的姓名、年龄和所在班级。

【例 3】查询院系编号为 34 但不是 11 年入学的学生学号、学生姓名和所在班级。

```
select sid,sName,sClassName
from student
```

```
    where exists
    (select sName,sClassName
    from student
    where sid like '__34%' and sid not like '____11%' );
```

本例属于多列子查询。多列子查询指子查询返回结果是多列的查询。多列子查询适用于成对比较和非成对比较，以检查数据的正确性。通常只要查询的数据与子查询返回的结果一一对应上了，查询才能成功；如果有一个没有对上，则查询不成功。“exists”是存在量词，当使用“exists”时，子查询不返回给外查询任何数据，只产生逻辑真“true”或者逻辑假“false”。使用“exists”的目标是判断非空等是否成立。

【例 4】查询学号为“1234120101”的学生姓名及平均分。

```
    select sname,(select AVG(totalScore) from score where sid='1234120101')
    from student
    where sid='1234120101';
```

如果按照子查询出现的位置划分，子查询可分为目标列子查询、内联表子查询、where 条件表达式子查询（例 1～例 3）、排序子查询、having 条件表达式子查询等。本列属于目标列子查询实例。

【例 5】查询学生姓名以及其选修课程的门数。

```
    select sname,scount
    from student,(select sid,COUNT(*) scount from score group by sid) as temp1
    where student.sid=temp1.sid;
```

本例属于内联表子查询实例。

【例 6】按学生总成绩升序输出学生姓名和所在班级。

```
    select sName,sClassName
    from student
    order by
    (select SUM(totalScore) from score where sid=student.sid)
```

本例属于排序子查询。

【例 7】查询平均成绩超过学号为“1234120101”的最高分的学生学号和平均成绩。

```
    select sid,AVG(totalScore)
    from score
    group by sid
    having   AVG(totalScore)>=(select   MAX(totalScore)   from   score   where
sid='1234120101');
```

本例属于 having 条件表达式子查询实例。

【例 8】子查询应用中常见错误。

```
    select sid,sName
    from student
    where sid=
    (select sid from score group by sid);
```

在本例中，子查询属于多行子查询，但应用了单行运算符，导致了错误的产生。子查询的另外常见错误是子查询没有返回结果，连接子查询的值和子查询中查询的内容不匹配，请读者自行举例。

应用 7　集合类查询

select 语句的查询结果是集合，所以对于 select 语句的查询结果可继续进行集合运算，主要有并（union）、交（intersect）、差（except）三个，但在执行集合类查询时各查询结果的列数必须相同，并且对应项的数据类型也必须相同。

【例 1】查询姓“刘”及性别是“女”的学生姓名、性别、出生年月等信息。

```
select sName,sSex,sBirthday from student where sName like '刘%'
union
select sName,sSex,sBirthday from student where sSex='女';
```

对于该查询，可以采用修改条件的方式替换为：

```
select sName,sSex,sBirthday from student
where sName like '刘%' or sSex='女'
```

【例 2】查询姓“李”与性别是“女”的学生姓名、性别、出生年月的交集。

```
select sName,sSex,sBirthday from student where sName like '李%'
intersect
select sName,sSex,sBirthday from student where sSex='女';
```

对于查询姓“李”与性别是“女”的学生姓名、性别、出生年月的交集，实质就是查询姓“李”并且性别是“女”的学生信息，因此，采用修改条件的方式替换 SQL 语句为：

```
select sName,sSex,sBirthday from student
where sName like '李%' and sSex='女';
```

【例 3】查询姓“刘”与电子邮箱中不带“4”的学生姓名、性别、出生年月的差集。

```
select sName,sSex,sBirthday from student where sName like '刘%'
except
select sName,sSex,sBirthday from student where sEmail not like '%4%';
```

同样，修改该查询条件后，替换的 SQL 语句为：

```
select sName,sSex,sBirthday from student
where sName like '刘%' and sEmail not like '%4%';
```

习题

基于本教材给出的学生成绩管理数据库，用 SQL 语句完成以下查询：

1. 查询学生表中的所有的党员的姓名，查询结果按学号升序排列。
2. 查询成绩表中综合成绩合格的所有成绩信息。
3. 查询学分为 3 或 4 的课程名称。

4. 查询所有课程名称中含有“程序”的课程号和课程名。
5. 查询所有姓王的学生信息。
6. 查询所有学生的姓名和出生年份，并给该出生年份属性列命名为出生年份。
7. 查询所有选课记录的课程号（不重复显示）。
8. 找出选修课程成绩最差的选课记录。
9. 查找成绩前 5 名的成绩信息。
10. 查询成绩为 85、90、95 和 100 分的学生的成绩信息。
11. 查询成绩在 90 到 100 分之间的所有的成绩信息。
12. 统计所有学生的平均年龄。
13. 查询所有选课的学生的学号及其综合成绩的平均分，按综合成绩平均分降序排列。
14. 查询至少选修了三门课程的学生姓名及其选修的门数。
15. 查询有三门以上成绩大于 90 分的学生的姓名。
16. 查询学号为 1234120104 的学生的姓名和选修的课程名及成绩。
17. 查询选修了“数据库系统原理”课程的学生学号和姓名。
18. 查询所有被选修的课程信息。
19. 查询选修“数据库系统原理”且成绩高于 90 分的学生姓名。
20. 查询选修学分为 3 或 4 且参加考试的学生学号、课程号。
21. 查询每个学生超过他选修课程平均成绩的课程名。
22. 查询没有选修“数据库系统原理”课程的学生姓名。
23. 查询选修了全部课程的学生姓名。

二、JDBC

JDBC 的英文全称是 Java DataBase Connectivity，中文名字为 Java 数据库连接，是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。JDBC 通常提供与数据库建立连接、发送、操作数据库的语句并处理结果三种操作。

JDBC 是面向对象的接口标准，一般由具体的数据库厂商提供。它的主要功能是管理存放在数据库中的数据，通过对象定义了一系列与数据库系统进行交互的类和接口。通过接口对象，应用程序可以完成与数据库的连接、执行 SQL 语句、从数据库中获取结果、获取状态及错误信息、终止事务和连接等。

JDBC 为 Java 程序提供统一、无缝地操作各种数据库的接口。因为实际应用中常常无法确定用户想访问什么类型的数据库，程序员使用 JDBC 编程时可以不关心它要操作的数据库是哪个厂家的产品，从而提高了软件的通用性。只要系统上安装了正确的驱动程序，JDBC 应用程序就可以访问其相关的数据库。

创建一个以 JDBC 连接数据库的程序，通常包含以下 6 个步骤。

1. 加载 JDBC 驱动程序

在连接数据库之前，首先要加载想要连接的数据库的驱动到 JVM（Java 虚拟机），这通过 Java.lang.Class 类的静态方法 forName(String className)实现。

例如：

```
    try{
    //加载 SQL Server 的驱动类
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
    }catch(ClassNotFoundException e){
    System.out.println("找不到驱动程序类 ，加载驱动失败！");
    e.printStackTrace() ;
    }
```

成功加载后，会将 Driver 类的实例注册到 DriverManager 类中。提供 JDBC 连接的 URL，连接 URL 定义了连接数据库时的协议、子协议、数据源标识。

书写形式：协议：子协议：数据源标识。

协议：在 JDBC 中总是以 jdbc 开始。

子协议：是桥连接的驱动程序或是数据库管理系统名称。

数据源标识：标记找到数据库来源的地址与连接端口。

例如：（SQL Server 的连接 URL）

```
"jjdbc:sqlserver://localhost:1433;databaseName=数据库名"
```

2. 创建数据库的连接

要连接数据库，需要向 Java.sql.DriverManager 请求并获得 Connection 对象，该对象就代表一个数据库的连接。

使用 DriverManager 的 getConnectin(String url , String username , String password)方法传入指定的欲连接的数据库的路径、数据库的用户名和密码来获得。

例如：

```
    //连接 SQL Server 数据库，用户名 sa,，密码是 hh123456
    String url = "jdbc:sqlserver://localhost:1433;databaseName=ssms" ;
    String username = "sa" ;
    String password = "hh123456" ;
    try{
   Connection con =
           DriverManager.getConnection(url , username , password ) ;
    }catch(SQLException se){
   System.out.println("数据库连接失败！");
   se.printStackTrace() ;
    }
```

3. 创建一个 Statement

要执行 SQL 语句，必须获得 Java.sql.Statement 实例，Statement 实例分为以下 3 种类型：

（1） Statement：由方法 createStatement 所创建。Statement 对象用于发送简单的 SQL 语句。

（2） PreparedStatement：由方法 prepareStatement 所创建。PreparedStatement 对象用于发送带有一个或多个输入参数（IN 参数）的 SQL 语句。PreparedStatement 拥有一组方法，

用于设置 IN 参数的值。执行语句时，这些 IN 参数将被送到数据库中。PreparedStatement 的实例扩展了 Statement，因此它们都包括了 Statement 的方法。PreparedStatement 对象有可能比 Statement 对象的效率更高，因为它已被预编译过并存放在那以供将来使用。

（3） CallableStatement：由方法 prepareCall 所创建。CallableStatement 对象用于执行 SQL 储存程序——一组可通过名称来调用（就像函数的调用那样）的 SQL 语句。CallableStatement 对象从 PreparedStatement 中继承了用于处理 IN 参数的方法，而且还增加了用于处理 OUT 参数和 INOUT 参数的方法。

不过通常来说 createStatement 方法用于简单的 SQL 语句（不带参数）、prepareStatement 方法用于带一个或多个 IN 参数的 SQL 语句或经常被执行的简单 SQL 语句，而 prepareCall 方法用于调用已储存过程。

例如：执行带两个参数的存储过程 demoSp。

```
        CallableStatement cstmt =
                            con.prepareCall("{CALL demoSp(? , ?)}") ;
```

4. 执行 SQL 语句

Statement 接口提供了三种执行 SQL 语句的方法：executeQuery、executeUpdate 和 execute。

（1） ResultSet executeQuery(String sqlString)：执行查询数据库的 SQL 语句，返回一个结果集（ResultSet）对象。

（2）int executeUpdate(String sqlString)：用于执行 insert、update 或 delete 语句以及 sql ddl 语句，如：create table 和 drop table 等。

（3） execute(sqlString)：用于执行返回多个结果集、多个更新计数或二者组合的语句。

具体实现的代码：

```
    ResultSet rs = stmt.executeQuery("SELECT * FROM ...") ;
     int rows = stmt.executeUpdate("INSERT INTO ...") ;
     boolean flag = stmt.execute(String sql) ;
```

5. 处理结果

两种情况：

（1） 执行更新返回的是本次操作影响到的记录数。

（2） 执行查询返回的结果是一个 ResultSet 对象。

ResultSet 包含符合 SQL 语句中条件的所有行，并且它通过一套 get 方法提供了对这些行中数据的访问。

使用结果集（ResultSet）对象的访问方法获取数据：

```
        while(rs.next()){
   String name = rs.getString("name") ;
    String pass = rs.getString(1) ; // 此方法比较高效，（列是从左到右编号的，并且从列 1 开始）
        }
```

6. 关闭 JDBC 对象

操作完成以后要把所有使用的 JDBC 对象全都关闭，以释放 JDBC 资源，关闭顺序和声明顺序相反：

（1） 关闭记录集。

（2） 关闭语句对象。

（3） 关闭连接对象。

```
  if(rs != null){   // 关闭记录集
try{
    rs.close() ;
}catch(SQLException e){
    e.printStackTrace() ;
}
  }
  if(stmt != null){   // 关闭语句对象
try{
    stmt.close() ;
}catch(SQLException e){
    e.printStackTrace() ;
}
  }
  if(conn != null){  // 关闭连接对象
 try{
    conn.close() ;
 }catch(SQLException e){
    e.printStackTrace() ;
 }
  }
```

尽管 JDBC 在 Java 语言层面实现了统一，但不同数据库仍旧有许多差异。为了更好地实现跨数据库操作，于是诞生了 Hibernate 项目，Hibernate 是对 JDBC 的再封装，实现了对数据库操作更宽泛的统一和更好的可移植性，Hibernate 的具体使用读者可以查阅相关资料。

三、JSP 连接 SQL Server 2008 步骤

1. 修改 SQL Server 的认证模式

SQL Server 有两种身份验证方式，SQL Server 身份验证和 Windows 身份验证，下面介绍的连接方式是通过 SQL Server 身份验证连接方式实现的，在连接数据库之前必须保证 SQL Server 2008 R2 是采用 SQL Server 身份验证方式而不是 Windows 身份验证方式。如果在安装时选用了后者，则重新设置如下。

打开 SQL Server 2008 R2，右击“对象资源管理器”中最上面的“服务器”，选择属性→安全性，会弹出如图 3-39 所示的“服务器属性”窗口，在“服务器属性”窗口右边框中选择“SQL Server 和 Windows 身份验证模式”，最后单击“确定”按钮。

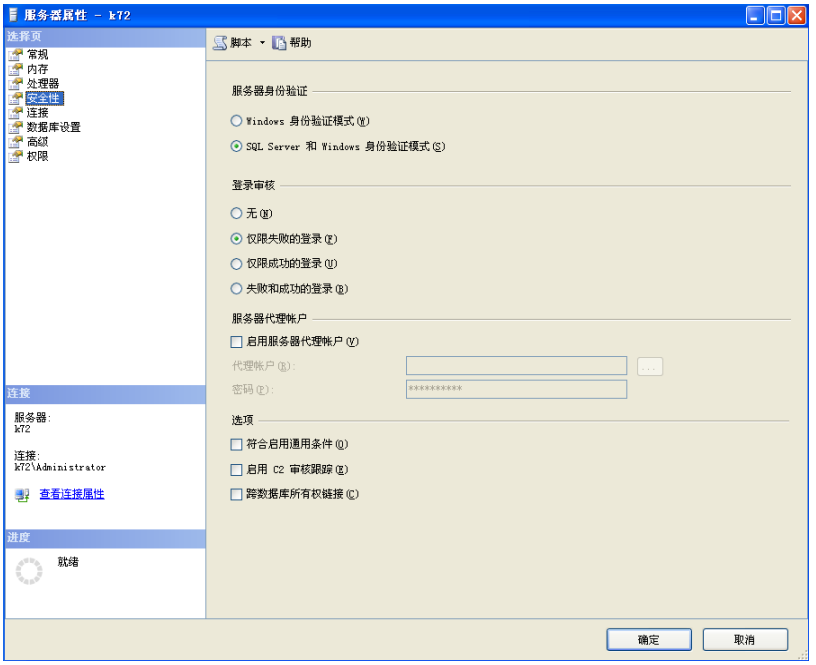


图 3-39　服务器属性窗口

保证 SQL Server 2008 R2 是采用 SQL Server 身份验证方式下开始 SQL Server 2008 R2 的配置。

2.　SQL Server 2008 R2 的配置

（1）　开始 → 所有程序 → Microsoft SQL Server 2008 R2→ 配置工具 → SQL Server 配置管理器，在打开的窗口的左边找到“SQL Server 网络配置”中的“MS SQLServer 的协议”，在窗口右边用鼠标右击“TCP/IP”，选择“已启用”。如果“Named Pipes” 未启用也设为启用。如图 3-40 所示。

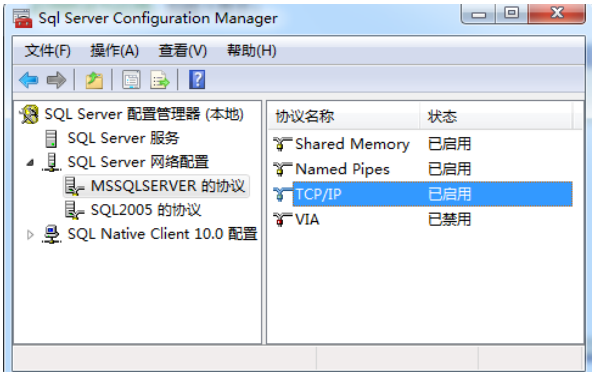


图 3-40　SQL Server Configuration Manager 窗口

（2）　鼠标右击图 3-40 右边的“TCP/IP”，在弹出的窗口中选择“IP 地址”标签，把 IP 地址中的“TCP 端口”设成“1433”，并将上方所有的“已启用”选项设置成“是”，如图 3-41 所示。如果有多个 IP 地址，选择“127.0.0.1”进行以上设置。

图 3-41　设置 TCP/IP 属性窗口

（3） 然后重启 SQL Server 服务器，重启完毕后，接下来要测试 1433 端口是否打开。

（4） 开始菜单 → 运行 cmd → 在命令提示符下输入“telnet 127.0.0.1 1433”，（注意 telnet 与 127 之间有空格，1 与 1433 之间有空格）若提示“不能打开到主机的连接，在端口 1433: 连接失败”，则说明 1433 端口没有打开，需要重新进行以上配置。若连接成功，显示如图 3-42 所示。

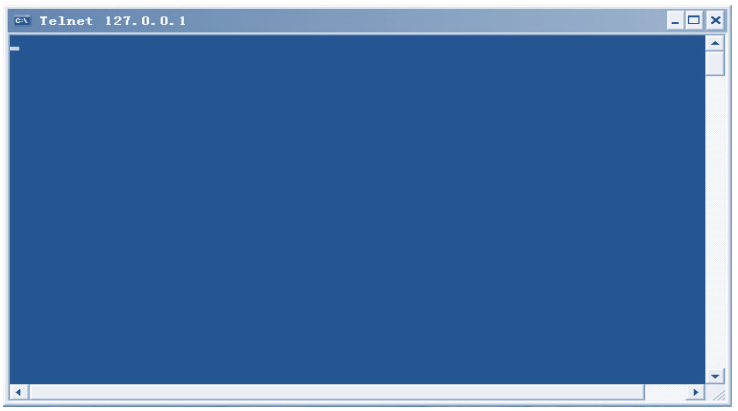


图 3-42　测试 1433 端口窗口

如果 telnet 不能使用，是因为操作系统的 telnet 指令没有打开，下面以 Win7 操作系统为例介绍打开 telnet 具体步骤如下。

① 首先进入系统“控制面板”，可以从开始菜单里找到“控制面板”，如图 3-43 所示：

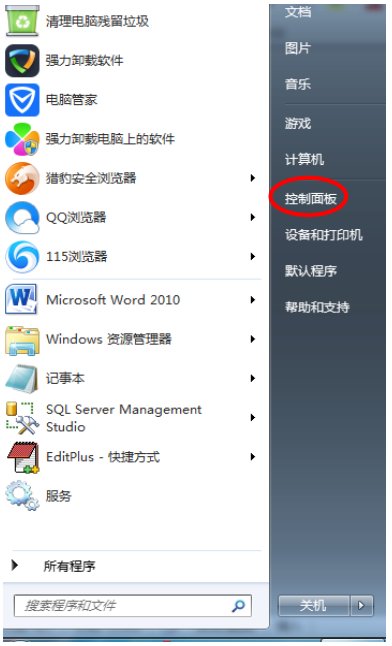


图 3-43　从开始菜单查找控制面板

② 进入控制面板后，找到“程序”如图 3-44 所示，双击进入，找到“程序和功能”，如图 3-45 所示。

图 3-44　控制面板中查找程序

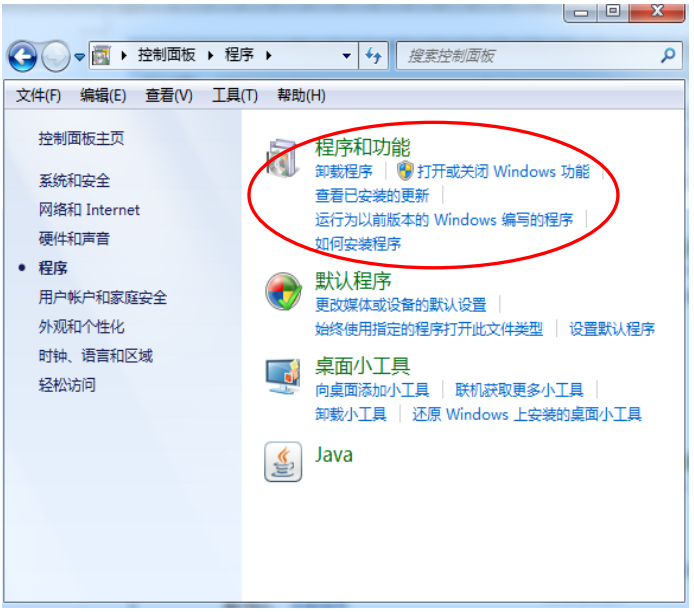


图 3-45　控制面板中的程序和功能

③ 双击“程序与功能”，则进入“程序与功能”面板，再在左侧找到“打开或关闭 Windows 功能”，并单击进入，如图 3-46 所示。

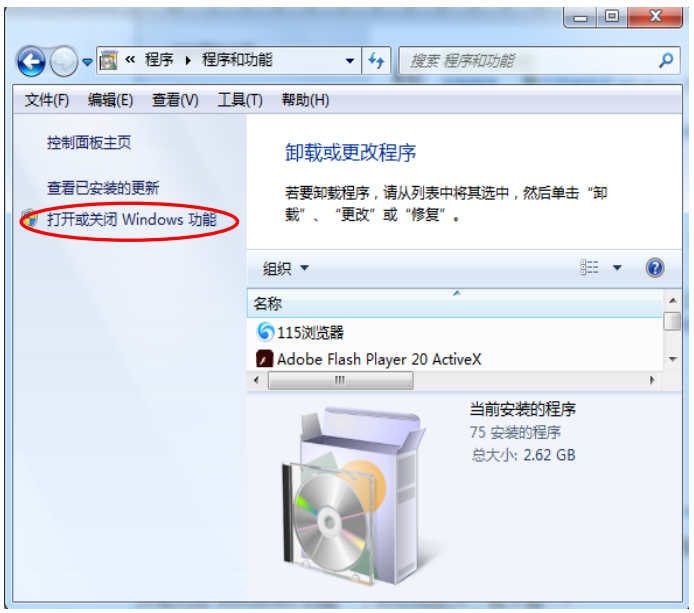


图 3-46　控制面板中打开或关闭 windows 功能

④ 最后一步，双击“打开或关闭 Windows 功能”会弹出“Windows 功能”对话框，在里面找到“Telnet 服务器”和“Telne 客户端”两项，然后将之前的勾选上即可开启对应服务了，打上勾后，要记得单击底部的“确定”按钮，如图 3-47 所示。

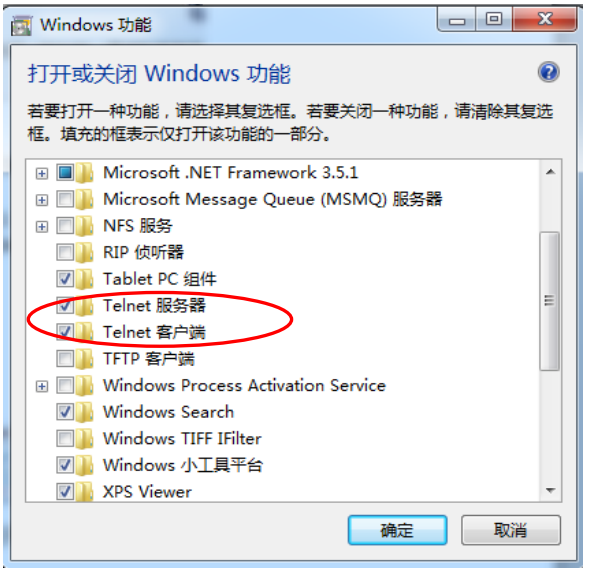


图 3-47　打开或关闭 windows 功能窗口

通过以上四步就可以轻松地开启 Win7 Telnet 服务了，之后可以进入 cmd 命令框中检测 telnet 是否开启成功。

3. 添加 SQL Server 驱动包

右键单击建立的工程名，依次选择：Build Path（构建路径）→ Configure Build Path（配置构建路径），在打开的窗口的右边选择“Libraries（库）”标签，然后单击“Add External JARs（添加外部 JAR）”，找到 sqljdbc4.jar 文件并打开，然后单击“确定”按钮完成构建路径的配置。如图 3-48 所示。

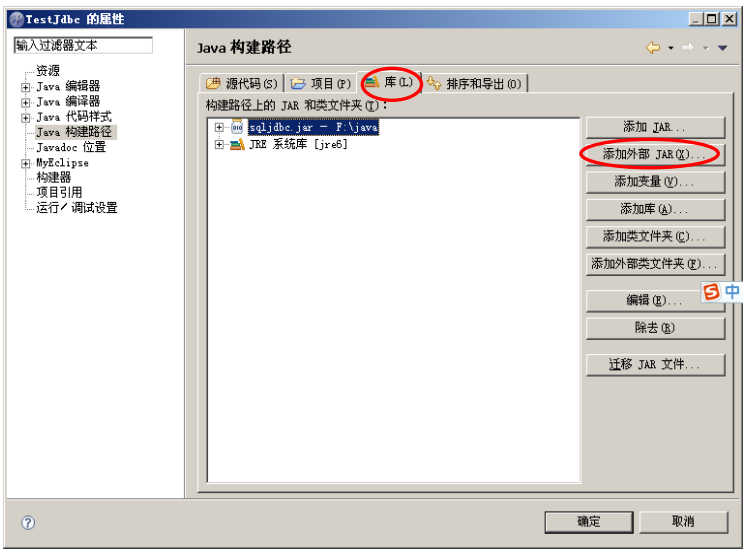


图 3-48　构建路径窗口

如果此方法加载驱动不成功则需要把 sqljdbc4.jar 放在 Tomcat 文件夹下的 lib 文件夹中，如本机的位置是 C:\Program Files\Apache Software Foundation\Tomcat 8.0\lib。如图 3-49 所示。

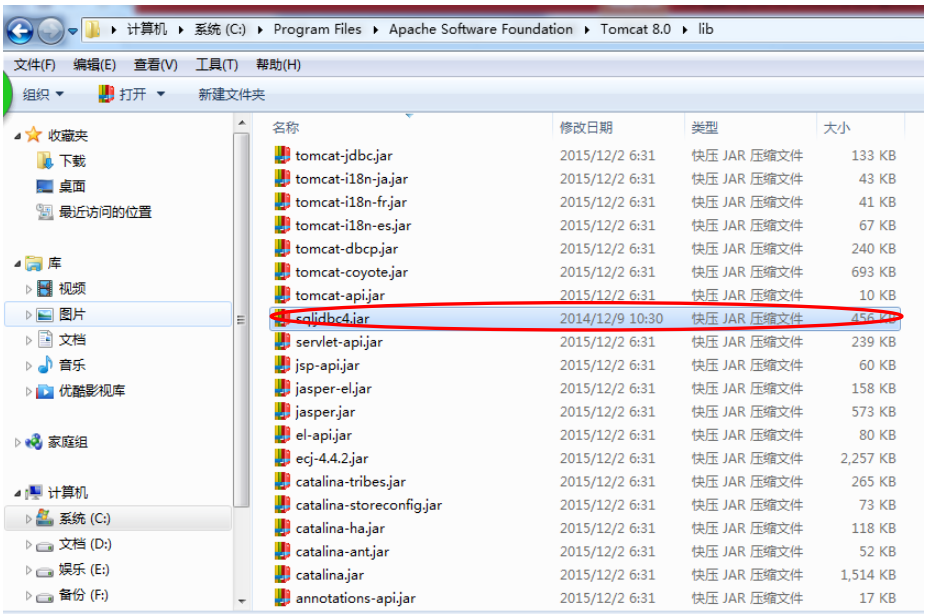


图 3-49　Tomcat 的 lib 目录

拓展知识

视图是由一个或者多个基本表组成的虚拟表，其内容由查询语句定义。视图和基本表一样，包含一系列带有名称的行、列数据，同样可以执行查询、修改、插入、删除操作（修改、插入、删除操作带有一定的限制）。但是，视图并不在数据库中以存储的数据子集形式存在。行和列数据来自由定义视图的查询所引用的表，并且在引用视图时动态生成。

视图的作用是应用简单、提供安全和保障逻辑数据独立性几个方面。应用简单是指在视图中看到的就是用户所需要的数据。提供安全是指视图约定用户的使用范围被限制在一定的数据子集上，为数据库提供一定的安全保证。逻辑数据独立性指视图对用户来说将基本表的结构屏蔽起来，以保障表结构变化对用户不带来影响。

一、视图与表的区别与联系

区别是：

（1） 视图是已经编译好的 SQL 语句，而表不是。

（2） 视图没有实际的物理记录，而表有。

（3） 表有具体的内容而视图是表的窗口。

（4） 表占用物理空间而视图不占用物理空间，视图只是逻辑概念的存在，表可以根据需要对它进行及时修改，但视图只能由创建的语句来修改。

（5） 表是内模式，视图是外模式。

（6） 视图是查看数据表的一种方法，可以查询基本表中某些列构成的数据，只是一些 SQL 语句的集合。从安全的角度说，视图可以不给用户接触基本表的机会，从而不知道基本表的内部结构。

（7） 表属于全局模式中的表，是实表；视图属于局部模式的表，是虚表。

（8） 视图的建立和删除只影响视图本身，不影响对应的基本表，反之，基本表删除后，虽然视图的定义仍然存在，但已经不能使用了。

联系是：

（1） 视图（view）是在基本表之上建立的表，它的结构（即所定义的列）和内容（即所有数据行）都来自基本表，它依据基本表存在而存在。

（2） 一个视图可以对应一个基本表，也可以对应多个基本表。

（3） 视图是基本表的抽象和在逻辑意义上建立的新关系。

二、视图定义

SQL 语句中，视图定义的一般格式为：

```
create view <视图> [(<列名>, [<列名>]…)]
as <子查询>
[with check option]
```

定义可更新视图时加上 with check option 短语，表示强制在视图上的所有数据更新语句都必须符合由 select 查询语句所设置的准则。

按照微软对视图的划分，视图分为标准视图、索引视图和分区视图。此处仅阐述标准视图，对于索引视图和分区视图，请读者自行参考相关资料。按照视图数据来源分为单表视图与多表视图两种。下面按照单表视图、多表视图的定义、查询、更新的方式逐一介绍。

1. 单表视图应用

【例 1】定义视图 v11 用于呈现课程编号为“3341201311”的成绩信息。

```
create view v11
as
select * from score where cid='3341201311';
```

本例定义视图 v11。

【例 2】基于视图 v11 定义视图 v1101 用于呈现选修该课程前 10 名的学生学号和总成绩。

```
create view v1101(id,grade)
as
select top 10 sid,totalScore from v11 order by totalScore desc;
```

【例 3】定义视图 v12 用于呈现信息工程学院学生的学号、姓名、性别、年龄信息，并且具备当对学生信息表进行修改或者插入操作时该视图能与学生信息表中数据同步。

```
go
create view v12(id,name,sex,age)
as
(select student.sid,sName,sSex,year(getdate())-year(sBirthday)
from student,department
where dName='信息工程学院' and substring(student.sid,3,2)=department.did)
with check option;
go
```

【例 4】查询视图 v11 得到选修该课程的所有学生学号和课程总成绩，结果以成绩降序排序。

```
select sid,totalScore from v11 order by totalScore desc;
```

【例 5】查询视图 v1101 得到前 3 名的学生学号、姓名和成绩。

```
select top 3 id,sName,grade from student,v1101
where sid=id;
```

【例 6】 采用课程表中成绩系数与视图 v11 中的成绩信息再次计算总成绩，对比视图 v11 中总成绩正确与否，将不正确成绩的学号给出。

```
select v11.sid from v11,
(select
sid,cast(cUsualResultsRatio*UsualResults+cExperimentalResultsRatio*Experime
ntalResults+cFinalAchievementRatio*examResults as int) as s
from course,v11 as myview where  course.cid=myview.cid)
as newScore
where (v11.sid=newScore.sid and v11.totalScore-newScore.s!=0);
```

本例利用视图生成校对数据，以便看出输入数据不一致之处。例如在超市管理中，常有进货数据、库存数据、销售数据，利用视图生成校对数据对查找商品。

定义视图后，对视图的查询没有什么限制，可以像对待表一样进行操作。但是，如果对视图中的元组进行更新操作（insert，update，delete）将受到限制。概括起来，关于可更新视图有以下三条规则：

（1） 若视图是基于多个表使用连接操作而导出的，那么对这个视图执行更新操作时，每次只能影响其中的一个表。

（2） 若视图导出时包含有分组和聚合操作，则不允许对这个视图执行更新操作。

（3） 若视图是从一个表经选择、投影而导出的，并在视图中包含了表的主键字或某个候选键，这类视图称为“行列子集视图”。对这类视图可执行更新操作。

【例 7】向视图 v12 中插入一个新学生的信息：学号“1234120121”、密码“120121”、姓名“燕青”、所在班级“软工 1201B”。

```
insert into v12(id,psw,name,class)
values('1234120122','120122','燕青','软工 1201B');
```

【例 8】向视图 v11 中插入一名学生选修课程号为 3341201311 的课程的成绩信息，其中，学号为“1234120122”、平时成绩为 50、实验成绩为 70、期末考试成绩为 90。

```
insert into v11(sid,cid,usualResults,examResults,experimentalResults)
values('1234120122','3341201311',50,70,90);
```

【例 9】将视图 v11 中学号为“1234120122”的学生平时成绩修改为 90。

```
update v11 set usualResults=90
where sid='1234120122';
```

【例 10】将视图 v12 中学号为“1234120101”的学生性别改为“男”。

```
update v12 set sex='男'
where id='1234120101';
```

【例 11】将视图 v12 中学号为“1234120122”学生的信息删除。

```
delete from v12 where id='1234120122';
```

2. 多表视图应用

对于多表视图，如果视图没有 instead of 触发器，或者视图不是分区视图，则视图只有满足下列条件才可更新：

（1） select 语句在选择列表中没有聚合函数，也不包含 top，group by，union（除非视图是分区视图）或 distinct 子句。聚合函数可以用在 from 子句的子查询中，只要不修改函数返回的值。

（2） select 语句的选择列表中没有派生列。派生列是由任何非简单列表达式（使用函数、加法或减法运算符等）所构成的结果集列。

（3） select 语句中的 from 子句至少引用一个表。select 语句不能只包含非表格格式的表达式（即不是从表派生出的表达式）。

insert，update 和 delete 语句在引用可更新视图之前，也必须如上述条件指定的那样满足某些限制条件。只有当视图可更新，并且所编写的 update 或 insert 语句只修改视图的 from 子句引用的一个基表中的数据时，update 和 insert 语句才能引用视图。只有当视图在其 from 子句中只引用一个表时，delete 语句才能引用可更新的视图。

【例 1】定义多表视图 v13 用于呈现学生的学号、姓名、课程号、课程名、成绩等信息。

```
go
create view v13(sid,sname,cid,cname,grade)
as
(select student.sid,sName,course.cid,cName,totalScore
from student,course,score where
student.sid=score.sid and course.cid=score.cid);
```

【例 2】查询多表视图 v13。

```
select * from v13 where grade>=90;
```

【例 3】向视图 v13 中插入一个学生的选课的成绩，其中学号为“1334120122”，姓名

“吴用”，课程号“3341201311”，课程名“数据库系统原理”，综合成绩 91。

```
   go
   create trigger tri_insert
   on v13
   instead of insert
   as
   begin
   declare @newid char(10),@newpsw varchar(20),@newname varchar(10),@class
varchar(30)
   declare @pr char(2),@cno char(2),@year char(2),@score int,@newcid
varchar(10)
   select @newid=sid,@newname=sName,@newpsw=sid from inserted
   select @newcid=cid,@score=grade from inserted
   set @year=substring(@newid,1,2);
   set @pr=substring(@newid,3,2);
   set @cno=substring(@newid,7,2);
   if(@pr='11')
   set @class='计科'
   else
   set @class='软工'
   set @class=@class+@year+@cno+'B'
   insert into student(sid,sName,sPassword,sClassName) values (@newid,
@newname,@newpsw,@class);
   insert into score(sid,cid,totalScore) values(@newid,@newcid,@score);
   end
   go
   insert into v13(sid,sname,cid,cname,grade)
   values('1334120122','吴用',' 3341201311','数据库系统原理',91)
   select * from v13;
```

【例 4】将视图 v13 中学号为“1334120122”的姓名改为“宋江”。

```
   go
   create trigger tri_update
   on v13
   instead of update
   as
   begin
   declare @newid char(10),@oldid char(10),@newname varchar(10)
   select @newid=sid,@newname=sName from inserted
   select @oldid=sid from deleted
   update student set sid=@newid,sName=@newname
   where sid=@newid;
   end
   go
   update v13 set sname='宋江' where sid='1334120122'
   select * from v13
   select * from student
```

【例 5】 将视图 v13 中“宋江”的选课信息删除。

```
go
create trigger Instead_tri
on v13
instead of delete
as
begin
declare @oldid char(10)
select @oldid=sid from deleted
delete score where sid=@oldid;
delete student where sid=@oldid;
end;
go
delete from v13 where sname='宋江'
select * from score
select * from student
```

案例 3　教师功能的设计与实现

案例 2 中我们设计并实现了教师用户登录功能，下面我们将继续实现教师功能中查询所授课程的成绩信息、其他类型的查询功能、增加学生成绩信息、修改学生成绩信息和删除学生成绩信息。

任务 1　查询所授课程的成绩信息

教师查看首界面“teacher.jsp”包括学号、姓名、课程名、班级、平时成绩、实验成绩、考试成绩、综合成绩、可执行的操作等信息，设计效果如图 3-50 所示。

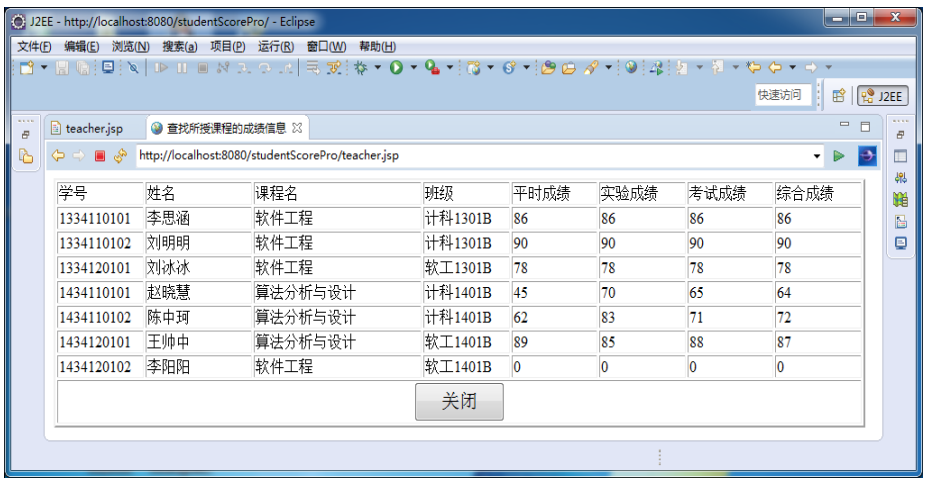


图 3-50　查询所授课程的成绩信息

其实现代码是在案例 2 的成功登录弹出提示信息框代码行

```
    out.print("<script> alert('登录成功! ') </script>");
```

后，加入语句

```
    location.href='teacher.jsp'</script>")";
```

该语句的作用是实现从登录界面页面跳转到教师界面页面 teacher.jsp 的跳转功能。

然后按照前面建立 JSP 页面的方法建立 teacher.jsp 页面。为了增加系统安全性，下面建立一个读取学生成绩信息的视图，创建该视图的代码如下：

```
    create view v stu score
    as
    select    student.sid,sName,course.cid,cname,tid,sClassName,usualResults,
experimentalResults,examResults ,totalScore
    from student,score,course
     where student.sid=score.sid
     and score.cid=course.cid
    teacher.jsp 页面代码如下：
    <%@page import="Java.sql.SQLException"%>
    <%@page import="Java.sql.ResultSet"%>
    <%@page import="Java.sql.Statement"%>
    <%@page import="Java.sql.Connection"%>
    <%@page import="Java.sql.DriverManager"%>
    <%@ page language="Java" contentType="text/html; charset=utf-8"
       pageEncoding="utf-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.
w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>查找所授课程的成绩信息</title>
    </head>
    <body>
       <form>
              <table border="2" width="900">
                     <tr>
                            <td width="80">学号</td>
                            <td width="100">姓名</td>
                            <td width="160">课程名</td>
                            <td width="80">班级</td>
                            <td width="80">平时成绩</td>
                            <td width="80">实验成绩</td>
                            <td width="80">考试成绩</td>
                            <td width="80">综合成绩</td>
                     </tr>
                     <%request.setCharacterEncoding("UTF-8");
                     response.setCharacterEncoding("utf-8");
                     String tid = session.getAttribute("uid").toString().trim();
                     String tpass = session.getAttribute("upass"). toString().
trim();
```

```
                        Connection conn = null;
                        Statement stmt = null;
                        ResultSet rs = null;
                        try {

      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
   conn   =   DriverManager.getConnection("jdbc:sqlserver://localhost:1433;
DatabaseName=ssms", "sa","hh123456");
                              stmt = conn.createStatement();
                        String sql = "select  *  from v_stu_score  where
tid="+tid;
                              rs = stmt.executeQuery(sql);
                              while (rs.next()) {%>
                  <tr>
                        <td><%=rs.getString(1)%></td>
                        <td><%=rs.getString(2)%></td>
                        <td><%=rs.getString(3)%></td>
                        <td><%=rs.getString(4)%></td>
                        <td><%=rs.getInt(5)%></td>
                        <td><%=rs.getInt(6)%></td>
                        <td><%=rs.getInt(7)%></td>
                        <td><%=rs.getInt(8)%></td>
                  </tr>
                  <%}
                        } catch (SQLException e) {
                  out.print("<script> alert('数据库操作出现异常！'+e);
</script>");
                        } finally {
                              if (rs != null)
                                    rs.close();
                              if (stmt != null)
                                    stmt.close();
                              if (conn != null)
                                    conn.close();
                        }%>
                  <tr>
                  <td colspan="8" align="center"><input id="close" type=
"button"
                              value="关闭" name="close"
                             style='font-size: 20px; width: 100px; height:
40px;'
                              onclick="window.close()"></td>
                  </tr>
            </table>
      </form>
   </body>
   </html>
```

习题

在案例 2 思考题的基础之上，实现学生用户登录后查询个人所选修全部课程的成绩信息。

任务 2　其他类型查询

在任务 1 的基础之上，添加其他类型的查询操作，例如按学号精确查询某位学生的成绩信息、按姓名模糊查询一批学生的成绩信息、某区间内的综合成绩信息等。

任务 2.1　按学号精确查询

在任务 1 的设计界面中，增加下拉列表，其默认值为“学号”，用户根据需求输入相应学生的学号进行精确成绩查询操作，其设计效果如图 3-51 所示。

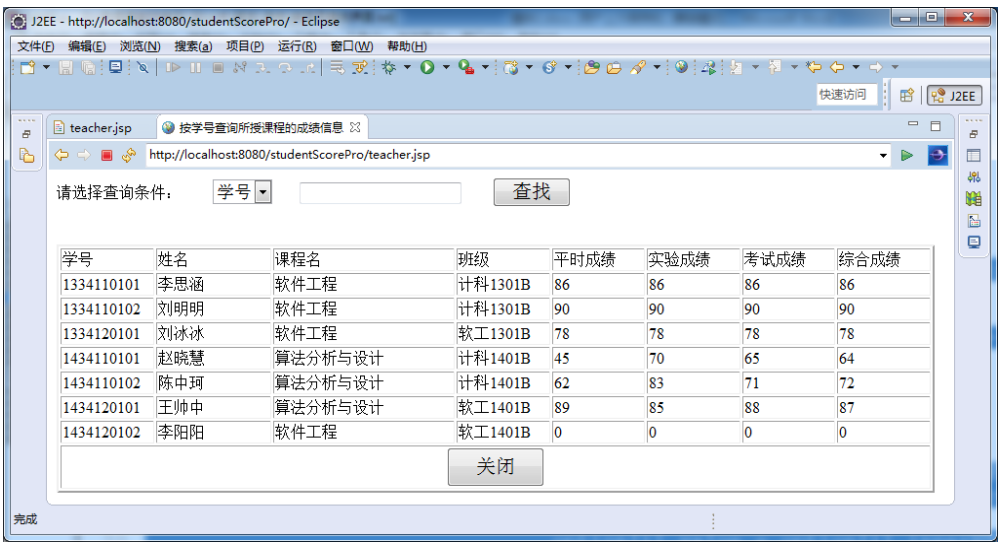

J2EE - http://localhost:8080/studentScorePro/ - Eclipse
文件(F) 编辑(E) 浏览(N) 搜索(a) 项目(P) 运行(R) 窗口(W) 帮助(H)
快速访问
J2EE
teacher.jsp
按学号查询所授课程的成绩信息
http://localhost:8080/studentScorePro/teacher.jsp
请选择查询条件：
学号
查找
学号 姓名 课程名 班级 平时成绩 实验成绩 考试成绩 综合成绩
1334110101 李思涵 软件工程 计科1301B 86 86 86 86
1334110102 刘明明 软件工程 计科1301B 90 90 90 90
1334120101 刘冰冰 软件工程 软工1301B 78 78 78 78
1434110101 赵晓慧 算法分析与设计 计科1401B 45 70 65 64
1434110102 陈中珂 算法分析与设计 计科1401B 62 83 71 72
1434120101 王帅中 算法分析与设计 软工1401B 89 85 88 87
1434120102 李阳阳 软件工程 软工1401B 0 0 0 0
关闭
完成

图 3-51　按学号精确查询设计效果图

按学号精确查询设计界面代码如下。

首先，在任务 1 的“<body>”标签后，加入如下代码：

```
<script language="Javascript">
        function change() {
            //下拉式列表框选项改变时读取下拉式列表框中的值
            var option = document.getElementById("select").options;
            for (var i = 0; i < option.length; i++)
                option[i].value = option[i].innerHTML;
            document.teacher.submit();//当前页面提交
        }
        function changeItem() {
            //改变下拉式列表框 select 时传递其选中的值，由 select 传递
            var selectitem = teacher.select.value;
            window.location.href = "teacher.jsp?select=" + selectitem;
```

```
            }
      </script>
      <%     request.setCharacterEncoding("utf-8");
             response.setCharacterEncoding("utf-8");
             String selectitem = "学号";//定义变量 selectitem 用于存放查找条件按学
号查找
             if (request.getParameter("select") != null)
                    selectitem = request.getParameter("select").toString().
trim();
      %>
```

接下来在“<form>”标签中加入如下属性：

```
   <form action=teacher.jsp name="teacher" method="post">
```

然后，在“<form>”标签后加入如下代码：

```
   请选择查询条件：      
   <select     style='font-size: 18px' name="select" onChange="changeItem()">
       <option value="学号" <%if (selectitem.equals("学号")) {%> selected <%}%>>
学号</option>
             </select>      
     <input type="text"    name="searchcontent" style="width: 160px" />
                    <input type="submit" value=
"查找"
                    style='font-size: 20px; width: 80px; height: 30px;' /><br>
<br>
```

修改完以上代码即可实现图 3-51 所示效果。

要实现按学号查找功能需要修改如下代码，修改语句“try”后与语句“rs = stmt.executeQuery(sql);”前的内容为：

```
   String selectcontent = null, select = null;
      if (request.getParameter("searchcontent") != null)
            selectcontent = request.getParameter("searchcontent").toString().
trim();
                    if (request.getParameter("select") != null)
                    selectitem = request.getParameter("select").toString().
trim();
                                  conn     =     DriverManager.getConnection
("jdbc:sqlserver://localhost:1433;DatabaseName=ssms", tid, tpass);
                                  stmt = conn.createStatement();
             String sql = "select  * from v_stu_score  where  tid="+ "'" + tid
+ "'";
                           if (!("".equals(selectcontent) || selectcontent ==
null)) {
                                  sql += "and student.sid='" + selectcontent +
"'";
                                  }
```

修改完以上代码即可实现按给定学号进行成绩查询的功能，实现如图 3-52 所示的按输入要查找的成绩信息的学生的学号，单击“查找”按钮即可实现如图 3-53 所示的查找功能。

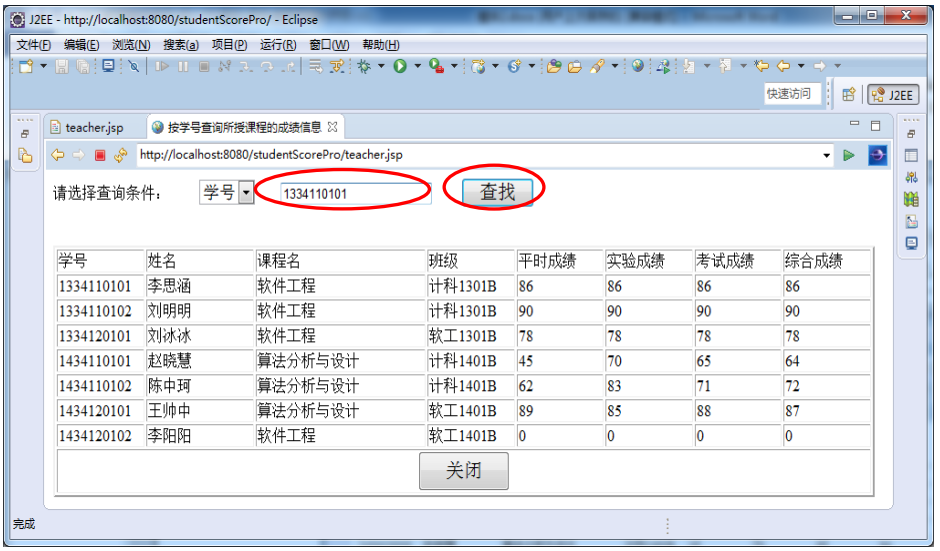

J2EE - http://localhost:8080/studentScorePro/ - Eclipse
文件(F) 编辑(E) 浏览(N) 搜索(a) 项目(P) 运行(R) 窗口(W) 帮助(H)
快速访问
J2EE
teacher.jsp
按学号查询所授课程的成绩信息
http://localhost:8080/studentScorePro/teacher.jsp
请选择查询条件：
学号
1334110101
查找
学号 姓名 课程名 班级 平时成绩 实验成绩 考试成绩 综合成绩
1334110101 李思涵 软件工程 计科1301B 86 86 86 86
1334110102 刘明明 软件工程 计科1301B 90 90 90 90
1334120101 刘冰冰 软件工程 软工1301B 78 78 78 78
1434110101 赵晓慧 算法分析与设计 计科1401B 45 70 65 64
1434110102 陈中珂 算法分析与设计 计科1401B 62 83 71 72
1434120101 王帅中 算法分析与设计 软工1401B 89 85 88 87
1434120102 李阳阳 软件工程 软工1401B 0 0 0 0
关闭
完成

图 3-52　输入学号按学号查找界面

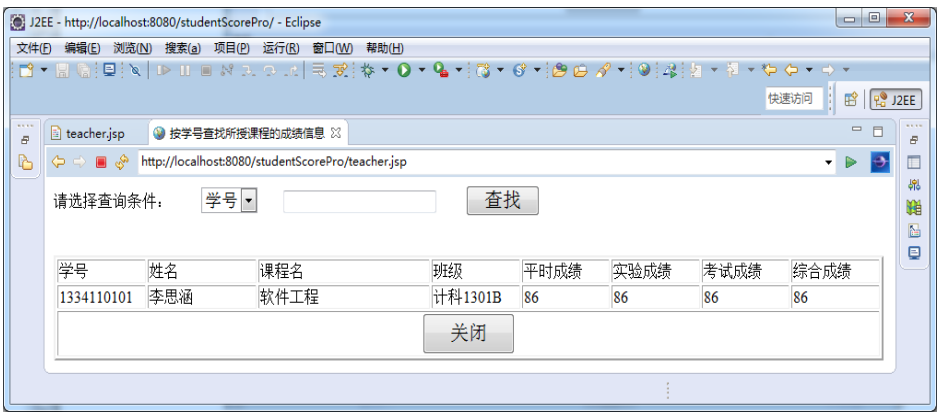


图 3-53　按学号查找结果图

任务 2.2　按姓名模糊或精确查询

在任务 2.1 的基础之上，对下拉列表中加入“姓名”查询项，以实现按姓名模糊或精确查询功能，其设计界面如图 3-54 所示。

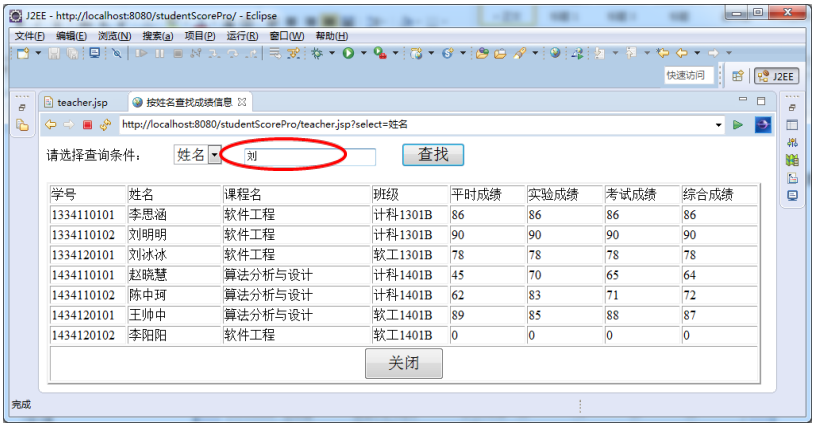


图 3-54　姓名模糊或精确查询设计界面

在图 3-54 下，输入姓“刘”，按下“查询”按钮，查询结果如图 3-55 所示。

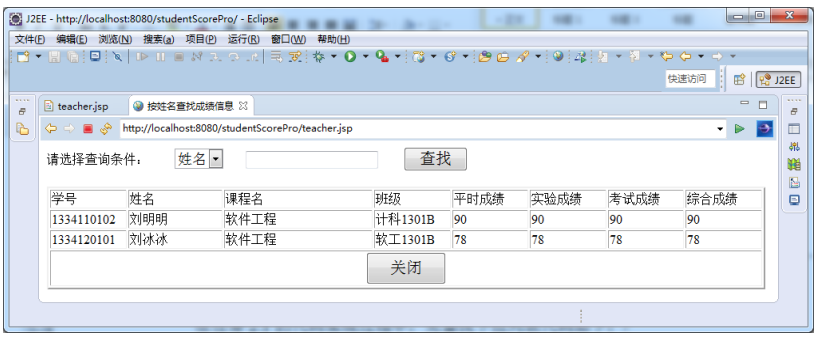


图 3-55　所授课程中姓“刘”学生的成绩信息

在任务 2.1 的代码实现过程中，先查找下拉列表代码如下：

```
    <option value="学号" <%if (selectitem.equals("学号")) {%> selected <%}%>>
学号</option>
```

在该代码的后面，加入如下代码：

```
    <option value="姓名" <%if (selectitem.equals("姓名")) {%> selected <%}%>>
姓名</option>
```

然后找到查询条件是否为空的代码如下：

```
    if(!("".equals(selectcontent)|| selectcontent==null))
    {sql+="and student.sid='" + selectcontent + "'";
    }
```

将其修改为：

```
    if(!("".equals(selectcontent)|| selectcontent==null))
              {if (selectitem.equals("学号"))
              sql += " and student.sid= '"+ selectcontent+ "'" ;
```

```
            else if (selectitem.equals("姓名"))
            sql += " and sName like '%" + selectcontent + "%'";
        }
```

任务 2.3　综合成绩查询

在任务 2.2 的基础之上，对下拉列表中加入“综合成绩”查询项，以实现按各种需要对综合成绩的查询，如精确查询、比较查询、范围查询、集合查询与无成绩查询等功能，其设计界面如图 3-56 所示。

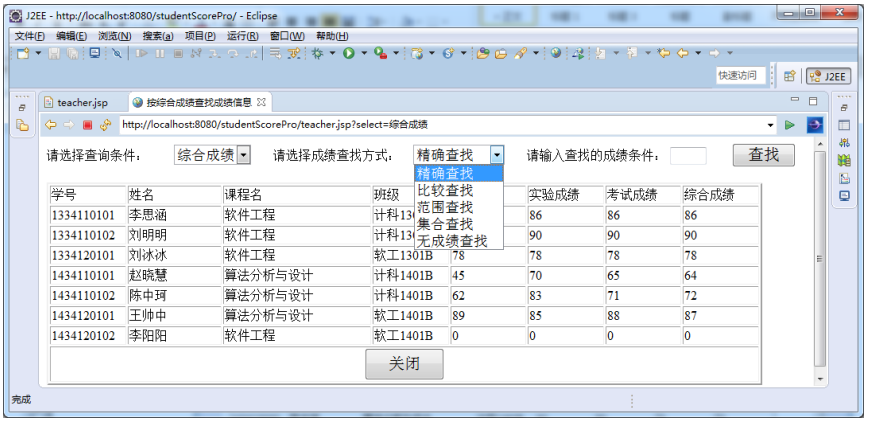


图 3-56　综合成绩查询设计图

该界面的设计过程如下。

在任务 2.2 的代码实现过程中，先查找代码

```
    String selectitem = "学号";
```

把该行代码改为

```
    String selectitem = "学号", selectscoreitem = "精确查找"
```

然后找到如下下拉列表代码：

```
    <option value="姓名" <%if (selectitem.equals("姓名")) {%> selected <%}%>>
姓名</option>
```

在这该行代码的后面，加入如下代码：

```
    <option value="综合成绩" <%if (selectitem.equals("综合成绩"))
    {%> selected<%}%>>综合成绩</option>
    在“<form>”标签后的
     <select ……</select>代码后、<input 代码前，加入如下代码：
    <% } else if (selectitem.equals("综合成绩")) {
              %>请选择成绩查找方式：     <select style='font-size: 18px'
                    name="selectscoretype" onChange="scorechangeItem()">
              <option value="精确查找" <%if (selectscoreitem.equals("精确查找"))
{%>
```

```
                              selected <%}%>>精确查找</option>
                <option value="比较查找" <%if (selectscoreitem.equals("比较查找"))
{%>
                              selected <%}%>>比较查找</option>
                <option value="范围查找" <%if (selectscoreitem.equals("范围查找"))
{%>
                              selected <%}%>>范围查找</option>
                <option value="集合查找" <%if (selectscoreitem.equals("集合查找"))
{%>
                              selected <%}%>>集合查找</option>
                <option value="无成绩查找" <%if (selectscoreitem.equals("无成绩查
找")) {%>
                              selected <%}%>>无成绩查找</option>
                </select>      
                <%      }
    if ((selectitem.equals("综合成绩")) && ((selectscoreitem.equals("精确查找"))
|| (selectscoreitem.equals("比较查找"))|| (selectscoreitem.equals("集合查找"))))
{%>
    请输入查找的成绩条件： <input type="text" name="searchscorecontent" width=200
                    style="width: 40px" />
                <%}
        if ((selectitem.equals("综合成绩")) && (selectscoreitem.equals("范围查
找"))) {%>
                查找成绩开始值<input type="text" name="beginscore" width=200
                    style="width: 40px" />     成绩
终止值<input   type="text" name="endscore" width=200 style="width: 40px" />
                <%}%>
```

以上代码完成综合成绩查询的设计界面。

1. 综合成绩精确查找

在综合成绩查询页面中，若选择成绩查询方式为精确查询，输入查找的成绩值，单击“查找”按钮可以实现按综合成绩的精确值进行查找，其设计效果如图 3-57 所示。

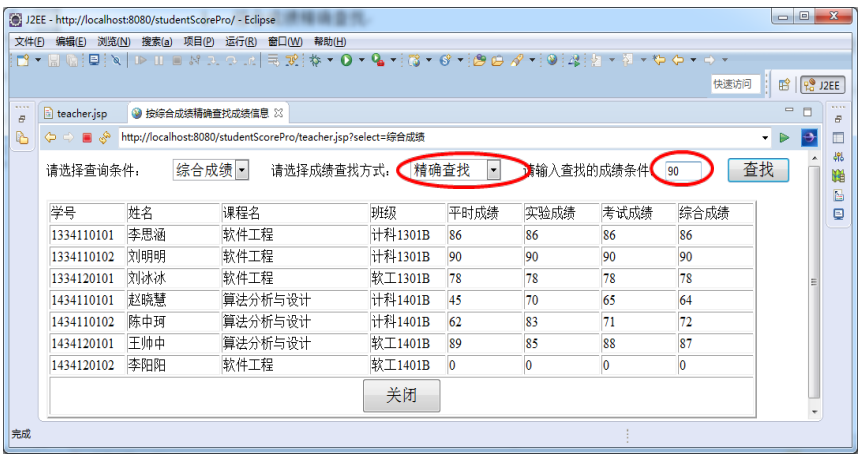


图 3-57　按综合成绩精确查找界面

在图 3-57 下，单击“查询”按钮，查询结果如图 3-58 所示。

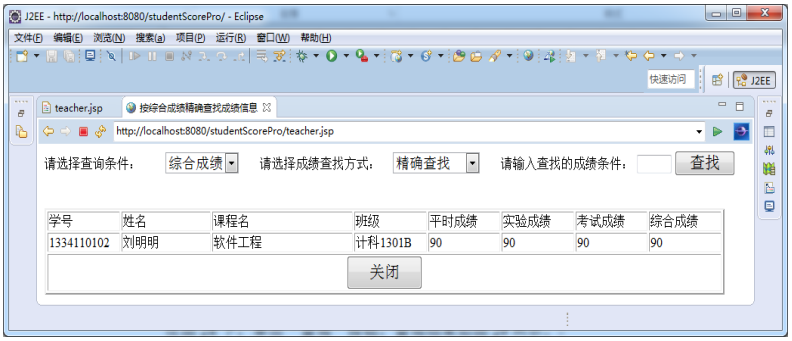


图 3-58　按综合成绩精确查找结果图

要实现以上按综合成绩精确查找，需要在上面综合成绩查找界面的代码中，把 Javascript 代码，即位于<script></script>标签内的代码用下面的代码替换：

```
       <script language="Javascript" charset="UTF-8" >
              function change() {
                     var option = document.getElementById("select").options;
                     var scoreoption = document.getElementById("selectscoretype").
options;
                     for (var i = 0; i < option.length; i++)
                            option[i].value = option[i].innerHTML;
                     for (var j = 0; j < scoreoption.length; j++)
                            scoreoption[j].value = scoreoption[j].innerHTML;
                     document.teacher.submit();
              }
              function changeItem() {
                     var selectitem = teacher.select.value;
                     window.location.href = "teacher.jsp?select=" + selectitem;
              }
              function scorechangeItem() {
                     var selectitem = teacher.select.value;
                     var selectscoreitem = teacher.selectscoretype.value;
                     window.location.href = "teacher.jsp?select=" + selectitem
                                   + "&selectscoretype=" + selectscoreitem;
              }
       </script>
```

然后在<form>标签前找到如下代码：

```
   if (request.getParameter("select") != null)
              selectitem=request.getParameter("select").toString().trim();
```

在该行代码的后面加入如下代码：

```
   if (request.getParameter("selectscoretype") != null)
   selectscoreitem   =   request.getParameter("selectscoretype").toString().
trim();
```

最后再找到下面代码：

```
String sql = "select  * from v_stu_score  where  tid="    + "'" + tid + "'";
```

把位于该行代码后，“rs = stmt.executeQuery(sql);”代码前中间的一段代码替换为如下代码：

```
   if (request.getParameter("select") != null) {
      if (selectitem.equals("学号") || selectitem.equals("姓名")) {
            if (!("".equals(selectcontent) || selectcontent == null)) {
                  if (selectitem.equals("学号"))
                        sql += "and student.sid='" + selectcontent + "'";
                  else if (selectitem.equals("姓名"))
                        sql += "and sName like '%" + selectcontent + "%'";
                                          }
                                    }
   else if (selectitem.equals("综合成绩")) {
            String searchscorecontent = null;
                  if (request.getParameter("searchscorecontent") != null)
   searchscorecontent = request.getParameter("searchscorecontent").toString().
trim();
                  if (!("".equals(searchscorecontent) || searchscorecontent
== null)) {
                  if (selectscoreitem.equals("精确查找"))
                  sql += "and totalScore=" + searchscorecontent;
                                          }
                                    }
                              }
```

修改完以上代码即可实现综合成绩的精确查找功能。

2. 综合成绩比较查找

在综合成绩查询页面中，若选择成绩查询方式为比较查询，输入查找的成绩的条件，单击“查找”按钮可以实现按综合成绩的比较查找，其设计效果如图 3-59 所示。

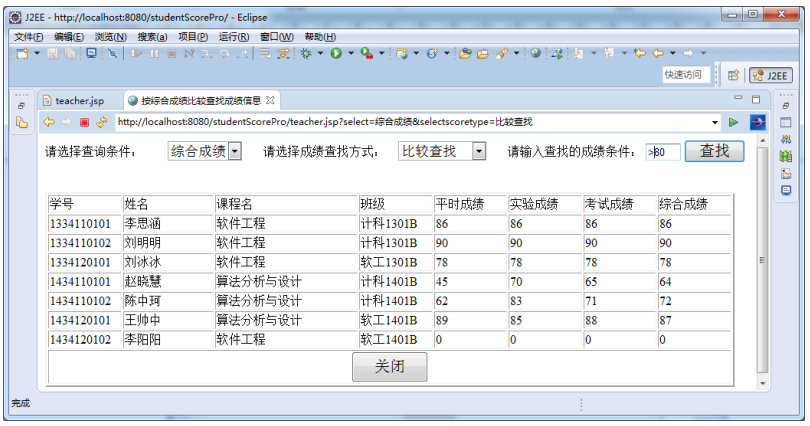


图 3-59　综合成绩比较查找界面图

在图 3-59 界面输入查找成绩条件，然后单击“查找”按钮，查询结果如图 3-60 所示。

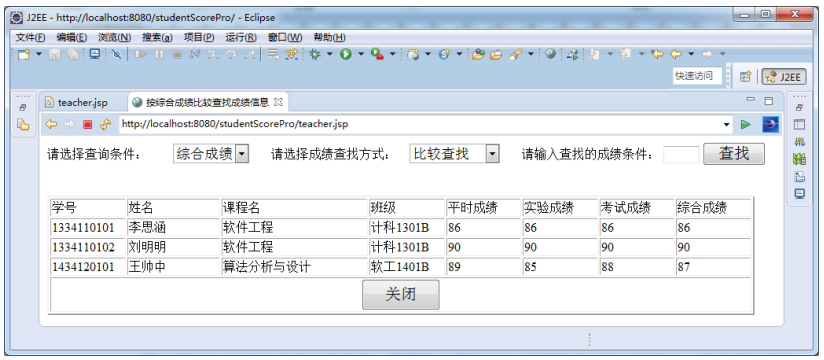


图 3-60　综合成绩比较查找结果图

要实现以上按综合成绩比较查找，需要在上面按综合成绩精确查找的代码中，查找到如下代码：

```
if (selectscoreitem.equals("精确查找"))
    sql += "and totalScore=" + searchscorecontent;
```

在该行代码后面添加如下代码：

```
else if (selectscoreitem.equals("比较查找"))
          sql += "and totalScore" + searchscorecontent;
```

修改完以上代码即可实现综合成绩的比较查找功能。

3. 综合成绩集合查找

在综合成绩查询页面中，若选择成绩查询方式为集合查询，输入集合查找的成绩条件，单击“查找”按钮可以实现按综合成绩的集合查找，其设计效果如图 3-61 所示。

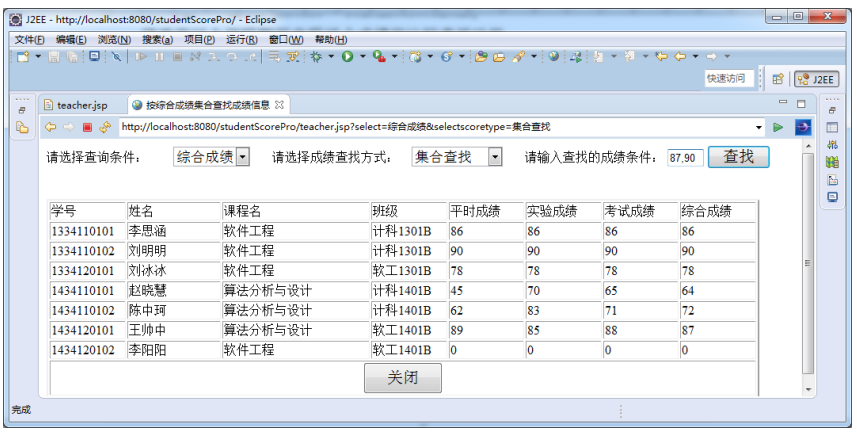


图 3-61　综合成绩集合查找界面图

在图 3-61 界面输入查找成绩条件，然后单击“查找”按钮，查询结果如图 3-62 所示。

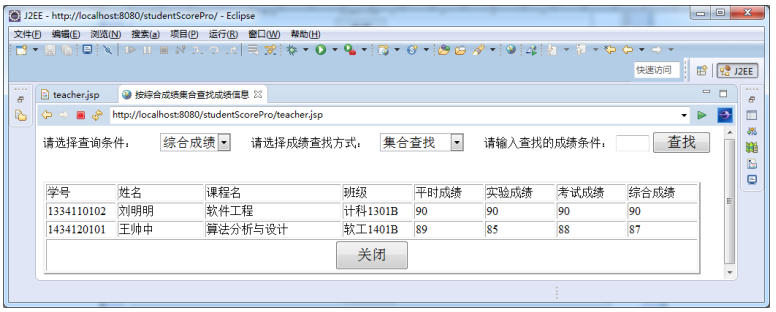


图 3-62　综合成绩集合查找结果图

要实现以上按综合成绩集合查找，需要在上面按综合成绩集合查找的代码中，查找到如下代码：

```
else if (selectscoreitem.equals("比较查找"))
            sql += "and totalScore" + searchscorecontent;
```

在该行代码后面加上如下代码：

```
else if (selectscoreitem.equals("集合查找"))
   sql += "and totalScore in (" + searchscorecontent + ")";
```

修改完以上代码即可实现综合成绩的集合查找功能。

4. 综合成绩无成绩查找

在综合成绩查询页面中，若选择成绩查询方式为无成绩查询，单击“查找”按钮可以实现按综合成绩的无成绩查找，查询效果如图 3-63 所示。

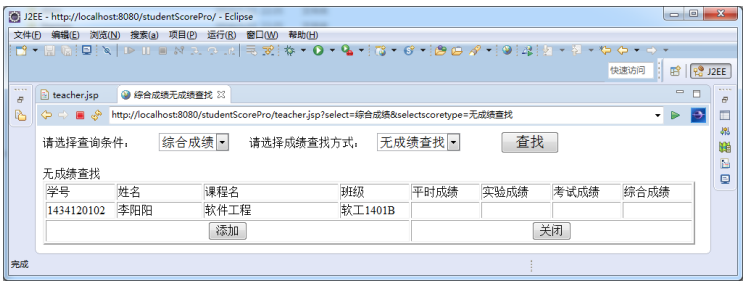


图 3-63　无成绩查找效果图

要实现以上按综合成绩无成绩查找，需要在上面按综合成绩集合查找的代码中，查找到如下代码：

```
else if (selectscoreitem.equals("集合查找"))
                sql += "and totalScore in (" + searchscorecontent + ")";
                                        }
```

在该行代码后面加上如下代码：

```
else if (selectscoreitem.equals("无成绩查找"))
          sql += "and totalScore is null";
```

然后再找到如下代码：

```
    rs = stmt.executeQuery(sql);
```

在该行代码之后和 while (rs.next())之前，加上如下代码：

```
    if (selectscoreitem.equals("无成绩查找") ) {
        String usualResults, examResults , experimentalResults , totalScore;
                while (rs.next()) {
    if (rs.getInt(5) == 0) usualResults= "";
              else   usualResults= String.valueOf(rs.getInt(5));
                     if (rs.getInt(6) == 0) examResults = "";
                     else   examResults= String.valueOf(rs.getInt(6));
                                 if (rs.getInt(7) == 0) experimentalResults= "";
                              else  experimentalResults=
String.valueOf(rs.getInt(7));
                               if (rs.getInt(8) == 0) totalScore = "";
                             else      totalScore=
String.valueOf(rs.getInt(8));
                     %>
                     <tr>
                            <td><%=rs.getString(1)%></td>
                            <td><%=rs.getString(2)%></td>
                            <td><%=rs.getString(3)%></td>
                            <td><%=rs.getString(4)%></td>
                            <td><%=usualResults%></td>
                            <td><%=examResults%></td>
                            <td><%=experimentalResults%></td>
                            <td><%=totalScore%></td>
                     </tr>
                     <%
                     }} else {
              最后在 catch (SQLException e) {
    out.print("<script> alert('数据库操作出现异常！'+e);</script>");代码前加上一个
“}”。
```

5. 综合成绩范围查找

若选择成绩查询方式为范围查找，则其设计如图 3-64 所示。

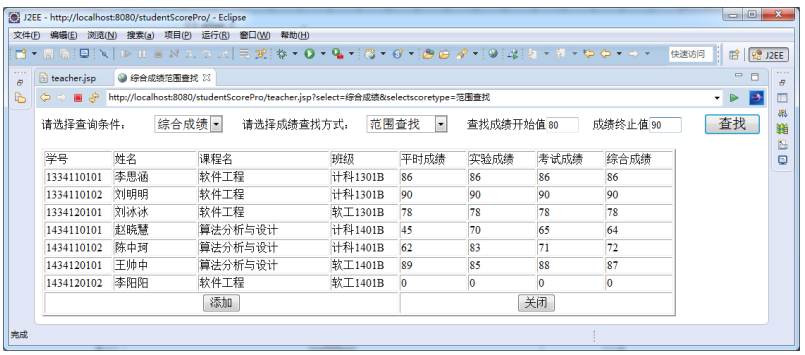


图 3-64　按综合成绩范围查找

在图 3-64 界面输入查找成绩的开始值和终止值，然后单击“查找”按钮，查询结果如图 3-65 所示。

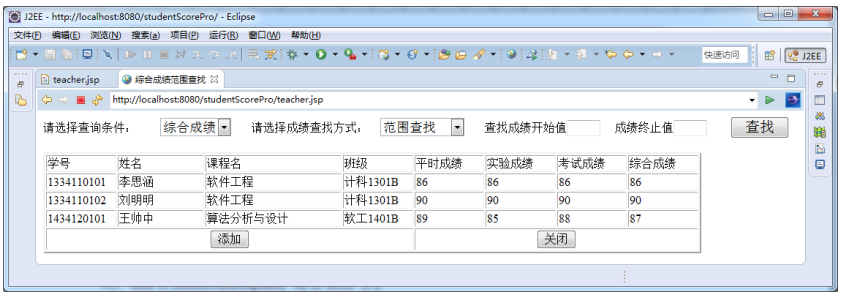


图 3-65　按综合成绩范围查找结果

要实现以上按综合成绩集合查找，需要在上面按综合成绩范围查找的代码中，查找到如下代码：

```
    else if (selectitem.equals("综合成绩")) {
                            、String searchscorecontent = null;
```

在该行代码后面加上如下代码：

```
    int beginscore = 0, endscore = 0;
```

然后再找到如下代码：

```
    else if (selectscoreitem.equals("无成绩查找"))
                            sql += " and  totalScore is null";
```

在该行代码后面加上如下代码：

```
                                    else if (selectscoreitem.equals("范围查找"))

       { if(request.getParameter("beginscore")!=null&&!("".equals(request.ge
tParameter("beginscore").toString().trim())))
       beginscore    =    Integer.valueOf((request.getParameter("beginscore").
toString().trim()));
       if(request.getParameter("endscore")!=null&&!("".equals(request.getPar
ameter("endscore").toString().trim())))
    endscore
=Integer.valueOf(request.getParameter("endscore").toString().trim());
                        if(beginscore!=0 && endscore!=0)
       sql += "and totalScore  between " + beginscore + " and " + endscore;
       }
```

修改完以上代码即可实现综合成绩的范围查找功能。

习题

在本案例任务 1 的基础之上，实现学生用户登录后查询个人成绩的精确、比较、范围、集合和无成绩查询功能。

任务 3　增加一个学生成绩

在任务 2 的基础之上，增加一个“添加”按钮实现增加一个学生成绩的功能。设计效果如图 3-66 所示。

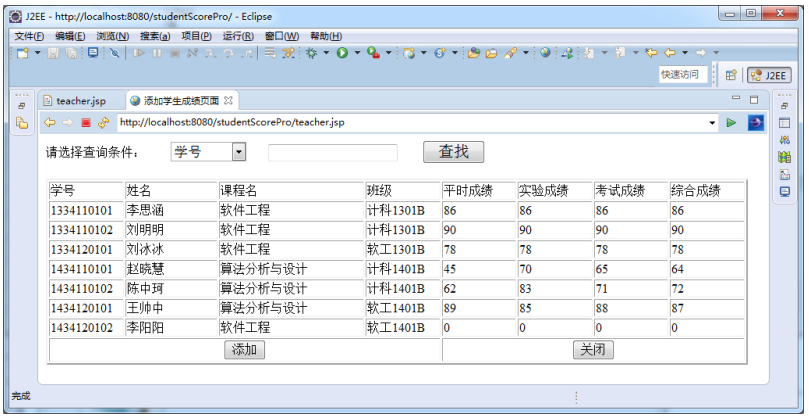


图 3-66　添加学生成绩界面

要实现以上添加学生成绩功能，需要在任务 2 的代码中，查找到如下代码“</table>”即 table 标签的结束标记，把其上面<tr></tr>标签内的代码替换成如下代码：

```
<td colspan="4" align="center"><input type="button" value="添加"
                              style='font-size: 16px'  /></td>
<td colspan="4" align="center"><input id="close" type="button"
style='font-size: 16px' value="关闭" name="close"
   onclick="window.close()"></td>
```

单击“添加”按钮会弹出如图 3-67 所示的输入要添加的学生成绩窗口。

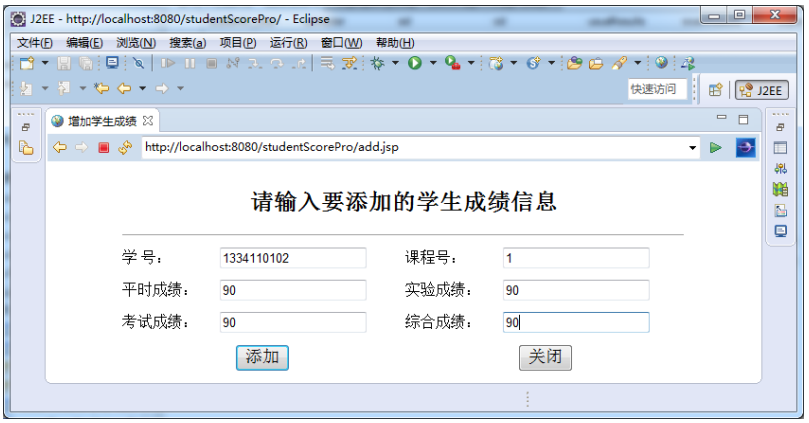


图 3-67　输入要添加的学生成绩信息窗口

要实现单击“添加”按钮进行页面跳转，跳转到增加学生成绩页面需要在上面添加命令按钮代码即“<input type="button" value="添加"style='font-size: 16px' />”的“/>”前加上如下代码：“onclick="window.location.href('add.jsp')"”，然后按照前面建立 JSP 文件的方

法，新建一个名为“add.jsp”的 JSP 文件，该文件的代码如下：

```
   <%@ page language="Java" contentType="text/html; charset=utf-8"
      pageEncoding="utf-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.
w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <title>增加学生成绩</title>
   </head>
   <body>
      <form name="add" method="post" action="addScore.jsp">
      <table  align="center" width="600">
      <tr>
      <td colspan="4" align="center">
            <h2>请输入要添加的学生成绩信息</h2>
            <hr>
            </td>
      </tr>
                  <tr>
                        <td  >学       号：</td>
                        <td  ><input  type="text"  name="addSid"  width=200
/></td>
                        <td  >课程号：</td>
                        <td><input  type="text"  name="addCid"  width=200
/></td>
                  </tr>
                  <tr></tr>    <tr></tr>
                  <tr>
                        <td>平时成绩：</td>
            <td><input type="text" name="addUsualResults" width=200 /></td>
                        <td>实验成绩：</td>
      <td><input type="text" name="addExperimentalResults" width=200 /></td>
                  </tr>
                  <tr></tr>    <tr></tr>
                  <tr>
                        <td>考试成绩：</td>
            <td><input type="text" name="addExamResults" width=200 /></td>
                        <td>综合成绩：</td>
            <td><input type="text" name="addTotalScore" width=200 /></td>
                  </tr>
                  <tr></tr>    <tr></tr>
                  <tr>
            <td colspan="2" align="center"><input type="submit" value="添加"
                              style='font-size: 18px' /></td>
            <td colspan="2" align="center"><input id="close" type="button"
                              style='font-size:  18px'  value=" 关 闭 "
name="close"
                              onclick="window.close()"></td>
```

```
                    </tr>
                </table>
            </form>
        </body>
        </html>
```

在图 3-67 输入要添加的学生成绩信息后单击“添加”按钮，会弹出一个提示添加成功的消息框如图 3-68 所示。

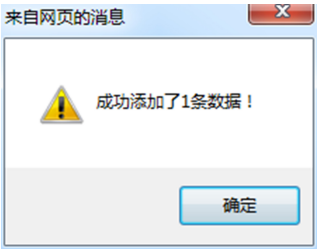


图 3-68　添加成功提示消息框

单击“添加”按钮实现添加功能，需要再建立一个实现向数据库添加数据的“addScore.jsp”页面，按照前面建立 JSP 页面的方法，建立一个名为“addScore.jsp”的 JSP 页面代码如下：

```
    <%@page import="Java.sql.SQLException"%>
    <%@page import="Java.sql.Statement"%>
    <%@page import="Java.sql.Connection"%>
    <%@page import="Java.sql.ResultSet"%>
    <%@page import="Java.sql.DriverManager"%>
    <%@page import="Java.sql.PreparedStatement"%>
    <%@ page language="Java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>添加成绩</title>
    </head>
    <body>
        <%      request.setCharacterEncoding("UTF-8");
                response.setCharacterEncoding("utf-8");
                PreparedStatement ps = null;
                Connection conn = null;
                Statement st=null;
                String tid = session.getAttribute("uid").toString().trim();
                String tpass = session.getAttribute("upass").toString().trim();
                String sid=null,cid=null;
                int
usualResults=0,examResults=0,experimentalResults=0,totalScore=0;
                if(request.getParameter("addSid")!=null)
                sid=new String(request.getParameter("addSid")).trim();
                if(request.getParameter("addCid")!=null)
                        cid=new String(request.getParameter("addCid")).trim();
                if(request.getParameter("addUsualResults")!=null)
    usualResults=Integer.parseInt(new
String(request.getParameter("addUsualResults")).trim());
                if(request.getParameter("addExamResults")!=null)
```

```
   examResults=Integer.parseInt(new
String(request.getParameter("addExamResults")).trim());
             if(request.getParameter("addExperimentalResults")!=null)
                   experimentalResults=Integer.parseInt(new
String(request.getParameter("addExperimentalResults")).trim());
             if(request.getParameter("addTotalScore")!=null)
   totalScore=Integer.parseInt(new
String(request.getParameter("addTotalScore")).trim());
             try{
      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
                   conn     =DriverManager.getConnection("jdbc:sqlserver://
localhost:1433;
   DatabaseName=ssms", "sa", "hh123456");
   String sql="select * from score where sid= '"+sid+"' and cid='"+cid+"' and
tid='"+tid+"'";
                   st=conn.createStatement();
                   ResultSet rs=st.executeQuery(sql);
   if(rs.next()==true) out.print("<script>alert('已经存在该生该课程的成绩!
');location.href='teacher.jsp'</script>");
                   else {sql="insert into score (sid,cid,usualResults,
examResults,experimentalResults,totalScore,tid) values(?,?,?,?,?,?,?)";
                ps=conn.prepareStatement(sql);
                   ps.setString(1, sid);
                   ps.setString(2, cid);
                   ps.setInt(3,usualResults);
                   ps.setInt(4,examResults);
                   ps.setInt(5,experimentalResults);
                   ps.setInt(6,totalScore);
                   ps.setString(7,tid);
                   int row=ps.executeUpdate();
                   if(row>0) out.print("<script>alert('成功添加了了 1 条数据!
');location.href='teacher.jsp'</script>");
                   else out.print("<script>alert('添加学生成绩失败! ');
location.href='teacher.jsp'</script>");
                   }
                   }
             catch(SQLException e){
   out.print("<script>alert('数据库异常 '+e);location.href='teacher.jsp'
</script>");
             }
             finally{if(ps!=null)
                   ps.close();
             if(conn!=null)
                   conn.close();
             }
      %>
      <br>
   </body>
   </html>
```

习题

在本案例任务 2 的基础之上，实现学生用户登录后添加一个选课记录的功能。

相关知识

insert 语句应用

insert 语句的格式是：

```
insert  into  <表名> [(<属性列 1> [，<属性列 2>] …)]
values ([<常量 1> [,<常量 2>]…]｜[子查询])
```

插入语句 insert 的含义为将常量数据或者子查询的结果插入到指定的表中。

注意：不管采用哪种数据插入方式、有多少数据要插入，但最终只能将数据插入到一个表中。

应用 1　完整单行数据的插入

【例 1】将信息工程学院（院系编号为 34）新增专业“物联网”、专业编号“080904”、修业年限为 4 年、学科门类“工学”、专业门类“计算机类”添加到专业表中。

```
insert into Professional(pid,pName,lengthOfSchooling,
disciplines,professionalCategory,did)
values('080904','物联网',4,'工学','计算机类','34');
```

或者

```
insert  Professional
values('080904','物联网',4,'工学','计算机类','34');
```

或者

```
insert  Professional(pid,pName,lengthOfSchooling,disciplines,
professionalCategory,did)
values('080904','物联网',4,'工学','计算机类','34');
```

以上三种方式均可实现查询要求，即将一条完整的数据插入到指定的表中，但第二种方式的使用要求是：

（1） 值集合中数据的次序与表中列同序；

（2） 值集合中数据的个数与表中列的个数相等；

（3） 值集合中数据的类型与表中列的类型相符。

注意：不要使用没有明确给出列的插入 insert 语句，尽量所有的插入语句 insert 都要明确给出相应的列名。

本例中第三种方式是省略 into 子句的情况，在当前流行的数据库管理系统下很多是支持的，但也有不支持的，因此，为防万一，使用插入语句 insert 时均带上 into 子句。

应用 2　单行部分数据的插入

【例 1】将信息工程学院（院系编号为 34）新增专业“物联网”、专业编号“080904”、专业门类“计算机类”添加到专业表中。

```
insert into Professional(pid,pName,professionalCategory,did)
values ('080904','物联网','计算机类','34');
```

或者

```
insert into Professional(pid,did,pName,professionalCategory)
values ('080904','34','物联网','计算机类');
```

当单行部分数据插入数据库时，必须遵守的原则是：

（1） 值集合中数据与插入语句中列的次序一致；

（2） 值集合中数据与插入语句中列的个数一一对应；

（3） 值集合中数据的类型与插入语句中列中要求的类型相符。

如果表的定义允许，则可以在插入 insert 语句中省略某些列，省略的列必须满足以下条件之一：

（1） 在表定义中给出默认值，这表示如果不给出值，将使用默认值；

（2） 在列定义时允许取空值。

本例中，修业年限 lengthOfSchooling 和学科门类 disciplines 均未在属性列上，由于在表定义时允许其取值为空，因此，当执行完该插入语句后，修业年限和学科门类取空值。

【例 2】将信息工程学院（院系编号为 34）新增专业“物联网”、专业编号“080904”添加到专业表中。

```
insert into Professional(pid,pName, professionalCategory,did)
values ('080904','物联网',null,'34');
```

本例是在列中给出专业门类，但在值集合中却没有给出，因此，必须在值集合中给一个空值“null”表示其缺省。

应用 3　多行数据的插入

【例 1】设有专业信息表 p1，该表结构与表 Professional 专业编号、专业名称和院系编号的结构一致，但数据不一致，需要将专业信息表 p1 内的数据归并到表 Professional 中。

```
insert into Professional(pid,pName,did)
select (pno,pName,dno) from p1;
```

本例是将两表合并的实例。两表有部分重复的表结构（列的类型及长度一样，但名称可以不同），将其中一个表的内容查询出来后插入到指定表中。

【例 2】生成课程编号是“3341201311”成绩总表 c11，包括学生学号、最终成绩。

```
create table c11
(id varchar(10) primary key,
```

```
    grade int );
    insert into c11(id,grade)
    (select sid,totalScore
    from score
    where cid='3341201311' );
```

本例首先创建用于存放课程“3341201311”的新表 c11，然后把所需要的数据批量插入到该表中。对于插入语句的执行过程是：先按照条件从成绩表 score 中读取一条记录，然后判断是否满足条件，如果满足条件，则直接取该条记录的学号 sid 和总成绩 totalScore 插入到新表 c11 中，否则舍弃后继续读取下一记录，直到将成绩表 score 的记录全部读完。

【例 3】要求同例 2，但要求采用存储过程完成上述要求。

```
    create proc p1
    as
    select sid,totalScore from score
    where cid='3341201311';
    go
    insert into c11(id,grade)
    execute p1;
```

本例首先创建存储过程 p1，然后通过插入语句将执行存储过程 p1 的结果全部插入到表 c11 中。

注意：插入语句 insert 中若采用子查询进行批量数据插入，则在子查询中不可以使用排序子句。

任务 4　修改学生成绩

在任务 3 的界面中，增加“修改”按钮，实现对相应学生成绩的修改功能，设计效果如图 3-69 所示。

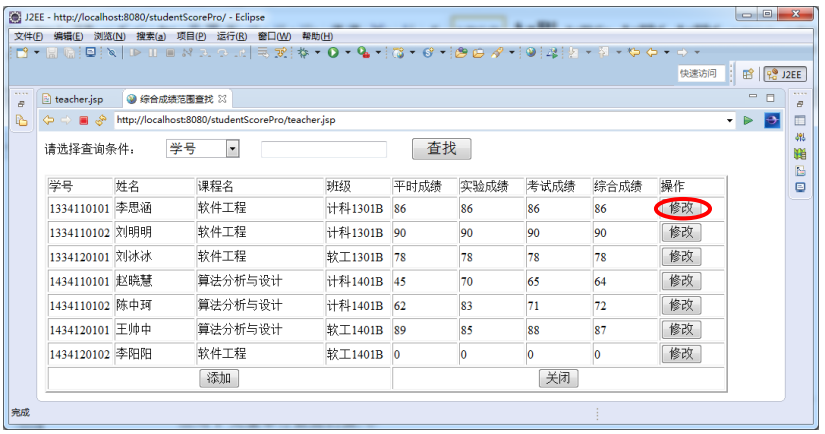

J2EE - http://localhost:8080/studentScorePro/ - Eclipse
teacher.jsp
综合成绩范围查找
http://localhost:8080/studentScorePro/teacher.jsp
请选择查询条件：
学号
查找
学号 姓名 课程名 班级 平时成绩 实验成绩 考试成绩 综合成绩 操作
1334110101 李思涵 软件工程 计科1301B 86 86 86 86 修改
1334110102 刘明明 软件工程 计科1301B 90 90 90 90 修改
1334120101 刘冰冰 软件工程 软工1301B 78 78 78 78 修改
1434110101 赵晓慧 算法分析与设计 计科1401B 45 70 65 64 修改
1434110102 陈中珂 算法分析与设计 计科1401B 62 83 71 72 修改
1434120101 王帅中 算法分析与设计 软工1401B 89 85 88 87 修改
1434120102 李阳阳 软件工程 软工1401B 0 0 0 0 修改
添加
关闭

图 3-69　修改学生成绩界面

要实现上面功能，需要在任务 3 插入功能实现的 teacher.jsp 页面代码中找到如下代码：

```
    <td width="80">综合成绩</td>
```

在其后加上下面代码：

```
    <td width="80">操作</td>
```

然后找到<td><%=totalScore%></td>和<td><%=rs.getInt(8)%></td>代码行，在上面两行代码的后面都加上如下代码：

```
    <td><input type="button" value="修改" style='font-size: 16px'

        onclick="window.location.href('update.jsp?sid=<%=rs.getString("sid")%
>&cname=<%=rs.getString("cname")%>')" /></td>
```

最后再找到如下代码：

```
    <td colspan="4" align="center"><input type="button" value="添加"
                                  style='font-size:  16px'  onclick="window.
location.href('add.jsp')" /></td>
```

把其中的数字 4 改为 5。

在某行数据后面单击“修改”按钮会弹出修改数据对话框如图 3-70 所示。

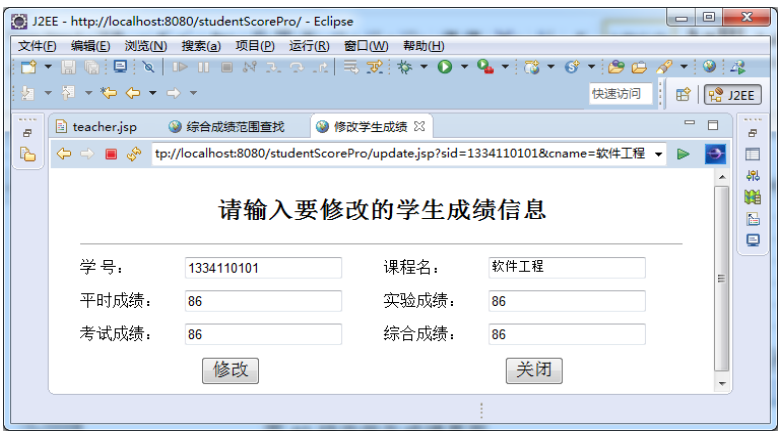


图 3-70　修改学生成绩信息页面

要实现单击图 3-70 中的“修改”按钮弹出图 3-71 修改学生成绩信息页面，需要添加 update.jsp 页面，按照建立 JSP 页面的方法，建立一个名为 update.jsp 的页面，该页面代码如下：

```
    <%@page import="Java.sql.ResultSet"%>
    <%@page import="Java.sql.Statement"%>
    <%@page import="Java.sql.Connection"%>
    <%@page import="Java.sql.DriverManager"%>
    <%@page import="Java.sql.PreparedStatement"%>
    <%@page import="Java.sql.SQLException"%>
    <%@ page language="Java" contentType="text/html; charset=utf-8"
```

```
       pageEncoding="utf-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <title>修改学生成绩</title>
   </head>
   <body>
      <form name="update" method="post" action="updateScore.jsp">
      <% request.setCharacterEncoding("UTF-8");
      response.setCharacterEncoding("utf-8");
      String sid=request.getParameter("sid");
      String cname=request.getParameter("cname");
      String tid = session.getAttribute("uid").toString().trim();
      String tpass = session.getAttribute("upass").toString().trim();
      int usualResults=0, experimentalResults=0,examResults=0,totalScore=0;
      Connection conn = null;
      Statement stmt = null;
      PreparedStatement ps=null;
      ResultSet rs = null;
      try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            conn = DriverManager.getConnection("jdbc:sqlserver://localhost:
1433;
   DatabaseName=ssms", "sa", "hh123456");
            stmt = conn.createStatement();
           String sql ="select usualResults,experimentalResults,examResults,
totalScore from score where sid='"+sid+"' and cid =(select cid  from course where
cname='"+cname+"')";
            rs = stmt.executeQuery(sql);
            while (rs.next()) {
                  usualResults=rs.getInt(1);
                  experimentalResults=rs.getInt(2);
                  examResults=rs.getInt(3);
                  totalScore=rs.getInt(4);
            }

      }catch (SQLException e) {
            out.print("<script> alert('数据库操作出现异常! '+e);</script>");
      } finally {
            if (rs != null)
                  rs.close();
            if (stmt != null)
                  stmt.close();
            if (conn != null)
                  conn.close();
      }
   %>
      <table  align="center" width="600">
```

```
      <tr>
      <td colspan="4" align="center">
            <h2>请输入要修改的学生成绩信息</h2>
            <hr>
            </td>
      </tr>
                  <tr>
                        <td  >学       号：</td>
   <td ><input type="text" name="updateSid" width=200  readOnly value=<%=sid%>
/></td>
                        <td  >课程名：</td>
   <td><input  type="text"  name="updateCname"  width=200   readOnly  value=<%=
cname%>/></td>
                  </tr>
            <tr> </tr><tr> </tr>
                  <tr>
                        <td>平时成绩：</td>
                        <td><input  type="text"  name="updateUsualResults"
width=200  value=<%=usualResults%> /></td>
                        <td>实验成绩：</td>
                        <td><input                              type="text"
name="updateExperimentalResults" width=200 value=<%=experimentalResults%> />
</td>
                  </tr>
                  <tr> </tr><tr> </tr>
                  <tr>
                        <td>考试成绩：</td>
                        <td><input  type="text"  name="updateExamResults"
width=200 value=<%=examResults%> /></td>
                        <td>综合成绩：</td>
                        <td><input   type="text"   name="updateTotalScore"
width=200 value=<%=totalScore%> /></td>
                  </tr>
                  <tr> </tr><tr> </tr>
                  <tr>
            <td colspan="2" align="center"><input type="submit" value="修改"
                              style='font-size: 18px' /></td>
      <td colspan="2" align="center"><input id="close" type="button"
                              style='font-size:  18px'  value=" 关 闭 "
name="close"
                              onclick="window.close()"></td>
                  </tr>
            </table>
      </form>
   </body>
   </html>
```

单击图 3-70 的“修改”按钮，弹出图 3-71 所示的“输入修改的成绩”信息窗口，修改后单击“修改”按钮，会弹出如图 3-72 所示的“成功修改学生成绩”的提示窗口，然后再跳转到查询所有学生成绩信息的 teacher.jsp 窗口。

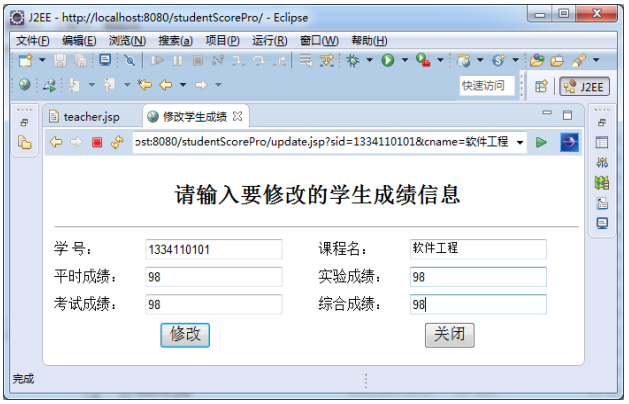


图 3-71　输入修改的学生成绩信息窗口

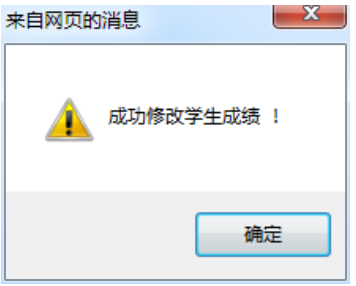


图 3-72　成功修改成绩提示窗口

要实现单击图 3-71 完成学生成绩的修改功能，需要建立 updateScore.jsp 页面，按照建立 JSP 页面的方法，建立名为 updateScore.jsp 的页面，该页面代码如下：

```
    <%@page import="Java.sql.SQLException"%>
    <%@page import="Java.sql.Statement"%>
    <%@page import="Java.sql.Connection"%>
    <%@page import="Java.sql.ResultSet"%>
    <%@page import="Java.sql.DriverManager"%>
    <%@page import="Java.sql.PreparedStatement"%>
    <%@ page language="Java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>修改学生成绩</title>
    </head>
    <body>
        <%      request.setCharacterEncoding("UTF-8");
                response.setCharacterEncoding("utf-8");
                PreparedStatement ps = null;
                Connection conn = null;
                Statement st = null;
                String tid = session.getAttribute("uid").toString().trim();
                String tpass = session.getAttribute("upass").toString().trim();
                String sid = null, cname = null;
                int usualResults = 0, examResults = 0, experimentalResults = 0,
totalScore = 0;
                if (request.getParameter("updateSid") != null)
                        sid = new String(request.getParameter("updateSid")).trim();
                if (request.getParameter("updateCname") != null)
                        cname = new String(request.getParameter("updateCname")).
trim();
```

```
            if (request.getParameter("updateUsualResults") != null)
                  usualResults   =   Integer.parseInt(new   String(request.
getParameter("updateUsualResults")).trim());
            if (request.getParameter("updateExamResults") != null)
                  examResults            =            Integer.parseInt(new
String(request.getParameter("updateExamResults")).trim());
            if (request.getParameter("updateExperimentalResults") != null)
                  experimentalResults = Integer
      .parseInt(new String(request.getParameter("updateExperimentalResults")).
trim());
            if (request.getParameter("updateTotalScore") != null)
                  totalScore    =    Integer.parseInt(new    String(request.
getParameter("updateTotalScore")).trim());
            if (usualResults < 0 || usualResults > 100) {%>
      <script>
            alert('输入的平时成绩超出成绩范围，请重新输入 ！ ');
            window.location.href = "update.jsp?sid=<%=sid%>&cname=<%=cname%>";
      </script>
      <%} else if (experimentalResults < 0 || experimentalResults > 100) {
      %>
      <script>
            alert('输入的实验成绩超出成绩范围，请重新输入 ！ ');
            window.location.href = "update.jsp?sid=<%=sid%>&cname=<%=cname%>";
      </script>
      <%     } else if (examResults < 0 || examResults > 100) {    %>
      <script>
            alert('输入的考试成绩超出成绩范围，请重新输入 ！ ');
            window.location.href = "update.jsp?sid=<%=sid%>&cname=<%=cname%>";
      </script>
      <%     } else if (totalScore < 0 || totalScore > 100) {
      %>
      <script>
                  alert('输入的总成绩超出成绩范围，请重新输入 ！ ');
   window.location.href = "update.jsp?sid=<%=sid%>&cname=<%=cname%>";
      </script>
      <%
            } else {
                  try {

      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
   conn    =    DriverManager.getConnection("jdbc:sqlserver://localhost:1433;
DatabaseName=ssms", "sa","hh123456");
                        conn.setAutoCommit(true);
                        String  sql  =  "update  score  set  usualResults=?,
examResults=?,experimentalResults=?,totalScore=? where sid=? and cid =(select
cid from   course where cname=?)";
                        ps = conn.prepareStatement(sql);
                        ps.setInt(1, usualResults);
                        ps.setInt(2, examResults);
                        ps.setInt(3, experimentalResults);
```

```
                            ps.setInt(4, totalScore);
                            ps.setString(5, sid);
                            ps.setString(6, cname);
                            int row = ps.executeUpdate();
                            if (row != 0)
   out.print("<script> alert('成功修改学生成绩! '); location.href='teacher.jsp'
</script>");
                      } catch (SQLException e) {
   out.print("<script> alert('修改学生成绩失败! '); location.href='teacher.jsp'
</script>");
                            e.printStackTrace();
                      } finally {if (ps != null)
                                  ps.close();
                            if (conn != null)
                                  conn.close();
                      }
               }
      %>
      <br>
   </body>
   </html>
```

习题

在本案例任务 3 的基础之上，实现学生用户登录后修改个人信息的功能。

相关知识

update 语句应用

```
update 语句称为修改语句，其语句的一般格式为：
update <表名>
set <列名>=<表达式> [,<列名>=<表达式>]…
[where <条件>]
```

该语句的作用是修改指定表中满足给定条件的列值，即通过 set 子句将指定列名的值替换成<表达式>的值。如果没有条件约束，则将修改指定表中全部行的数据。

在应用中，根据是否带有条件可分为有条件的修改语句和无条件的修改语句两种；根据 set 子句后列的情况可分为修改一列、修改多列和置空值三种情况；根据条件是否带子查询分为一般修改和带子查询的修改两种。由于多数教材都是按照有无条件的划分进行描述，故此处不再赘述。

应用 1　修改一列

【例 1】将所有课程的平时成绩加 10 分。

```
update score set usualResults=usualResults+10;
```

本例是无条件修改一列的情况，因此会将成绩表 score 中所有课程的平时成绩均加 10

分，在实际应用中，谨慎使用。

应用 2　修改多列

【例 1】将学号为“1234120101”、课程编号为“3341201311”的考试成绩和实验成绩分别加 10 分。

```
   update    score    set    examResults=examResults+10,experimentalResults=
experimentalResults+10
   where sid='1234120101' and cid='3341201311';
```

本例是有条件修改多列的情况，由条件学号和课程编号指定所修改的多列在一个数据行中，因此修改具有针对性。

应用 3　置空值

【例 1】将姓“刘”学生的考试成绩置空。

```
   update score set experimentalResults=null
   where sid in
   (select sid from student where sName like '刘%');
```

本例是带有子查询置空值的情况，修改语句的条件是由子查询构成。

注意：置空值使用确保该列的定义中允许取空值。

任务 5　删除学生成绩

在任务 4 的界面中，增加“删除”按钮，实现对对应学生成绩的删除功能，设计效果如图 3-73 所示。

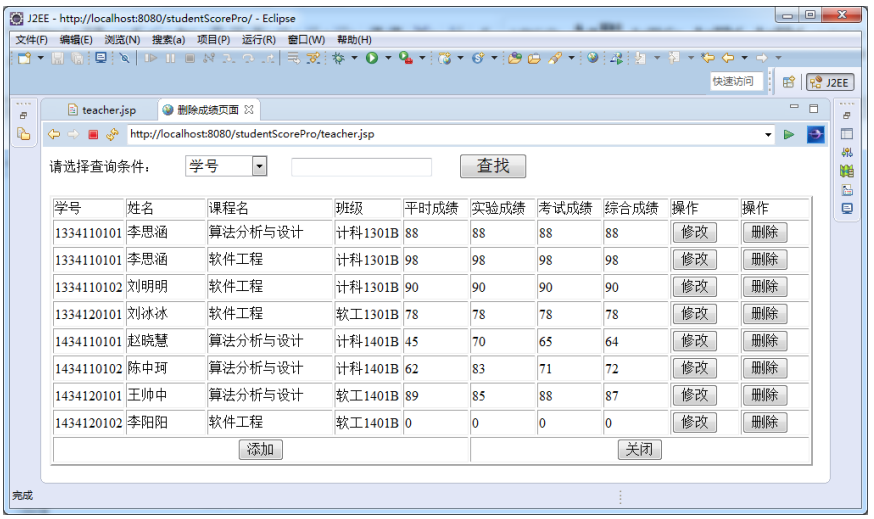

J2EE - http://localhost:8080/studentScorePro/ - Eclipse
文件(F) 编辑(E) 浏览(N) 搜索(a) 项目(P) 运行(R) 窗口(W) 帮助(H)
快速访问
J2EE
teacher.jsp
删除成绩页面
http://localhost:8080/studentScorePro/teacher.jsp
请选择查询条件：
学号
查找
学号 姓名 课程名 班级 平时成绩 实验成绩 考试成绩 综合成绩 操作 操作
1334110101 李思涵 算法分析与设计 计科1301B 88 88 88 88 修改 删除
1334110101 李思涵 软件工程 计科1301B 98 98 98 98 修改 删除
1334110102 刘明明 软件工程 计科1301B 90 90 90 90 修改 删除
1334120101 刘冰冰 软件工程 软工1301B 78 78 78 78 修改 删除
1434110101 赵晓慧 算法分析与设计 计科1401B 45 70 65 64 修改 删除
1434110102 陈中珂 算法分析与设计 计科1401B 62 83 71 72 修改 删除
1434120101 王帅中 算法分析与设计 软工1401B 89 85 88 87 修改 删除
1434120102 李阳阳 软件工程 软工1401B 0 0 0 0 修改 删除
添加
关闭
完成

图 3-73　删除学生成绩界面

单击其中一行数据对应删除操作会弹出如图 3-74 所示的确认删除提示框。单击消息提

示框中的“确定”按钮，会弹出如图 3-75 所示的成功删除数据消息框。

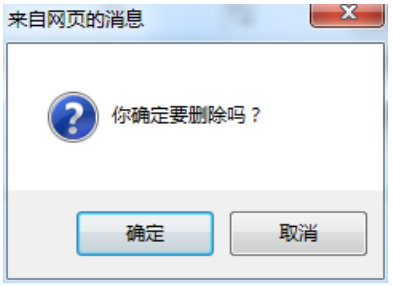


图 3-74　确认删除消息提示框

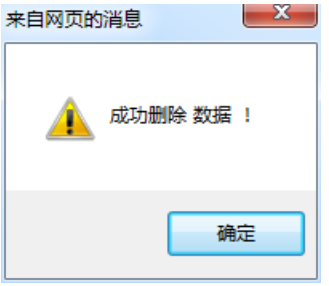


图 3-75　成功删除数据消息提示框

单击消息提示框中的“确定”按钮，确认进行删除操作，即可完成数据的删除操作。

要实现删除学生成绩信息，需修改以下代码在任务 4 的 teacher.jsp 页面找到 script 标签的结束标记即</script>，在其上添加如下代码：

```
function del(sid, cname) {
                if (confirm("你确定要删除吗？")) {
            window.location.href = "delete.jsp?sid=" + sid + "&cname="+ cname;
                }
    }
```

然后再找到如下代码：

```
<td width="80">操作</td>
```

在该行代码后面添加如下代码：

```
<td width="80">操作</td>
```

最后找到两行如下代码<td><input type="button" value="修改" style='font-size: 16px' onclick="window.location.href('update.jsp?sid=<%=rs.getString("sid")%>&cname=<%=rs.getString("cname")%>')" /></td>，在该行代码后加入如下代码：

```
<td><input type="button" value="删除" style='font-size: 16px'

    onclick="del('<%=rs.getString("sid")%>','<%=rs.getString("cname")%>')
" /></td>
```

单击“删除”按钮，要跳转到 delete.jsp 页面进行数据删除处理，按照建立 JSP 页面的方法，建立名为 delete.jsp 的 JSP 文件，该文件的代码如下：

```
<%@page import="Java.sql.PreparedStatement"%>
<%@page import="Java.sql.SQLException"%>
<%@page import="Java.sql.ResultSet"%>
<%@page import="Java.sql.Statement"%>
<%@page import="Java.sql.Connection"%>
<%@page import="Java.sql.DriverManager"%>
<%@ page language="Java" contentType="text/html; charset=UTF-8"
```

```
       pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>删除成绩</title>
   </head>
   <body>
   <% request.setCharacterEncoding("UTF-8");
   response.setCharacterEncoding("utf-8");
   String sid=request.getParameter("sid");
   String cname=request.getParameter("cname");
   Connection conn = null;
   PreparedStatement ps = null;
   try {Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
   conn    =    DriverManager.getConnection("jdbc:sqlserver://localhost:1433;
DatabaseName=ssms", "sa", "hh123456");
       String sql = "delete from score where sid=? and cid=(select cid from course
where cname=?)";
    ps=conn.prepareStatement(sql);
   ps.setString(1,sid);
   ps.setString(2,cname);
   if(ps.executeUpdate()!=0)
       out.print("<script> alert('成功删除数据 ! ');location.href='teacher.jsp'
</script>");
   else out.print("<script> alert('删除数据失败  ! ');location.href= 'teacher.
jsp'</script>");
   }
   catch(SQLException e){out.print("数据库操作异常 !  ");}
   finally{
       ps.close();
       conn.close();
   }
   %>
   </body>
   </html>
```

习题

在本案例任务 4 的基础之上，实现学生用户登录后删除选修课程的功能。

相关知识

delete 语句应用

delete 语句为删除数据语句，其语句格式为：

```
   delete from <表名>
   [where <条件>]
```

delete 语句的作用是从指定的表中删除满足给定条件的数据行。如果缺省条件，则表示删除表中所有数据，但表的结构不会被删除，即表的结构仍然存在。

在应用中，根据删除数据量的多少分为单行数据删除和多行数据删除；根据条件的有无及条件中是否有子查询分为带有子查询条件的删除、一般条件删除和无条件删除。

应用 1　无条件删除。

【例 1】删除课程成绩表 c11 中所有数据。

```
delete from c11;
```

本例是从课程成绩表 c11 中删除所有数据，但表 c11 的结构仍然存在，即语句 delete 仅删除表中数据，而不删除结构。

注意：由于数据删除后数据是不可恢复的，因此，采用该形式时一定要三思。

应用 2　一般条件删除

【例 1】删除学号为“1234120101”所选课程编号为“3341201311”的成绩信息。

```
delete from score where sid='1234120101' and cid='3341201311';
```

本例是单行数据的删除，由所给条件学号'1234120101' 和课程号'3341201311'指定一行数据。

【例 2】删除学号为“1234120101”所选全部课程信息。

```
delete from score where sid='1234120101';
```

本例是多行数据的删除。

应用 3　带有子查询条件的删除

【例 1】删除姓名为“刘冰冰”的全部选课信息。

```
delete from score where sid in
( select sid from student where sName='刘冰冰');
```

本例通过子查询的结果指定删除对象。

案例 4　密码修改界面的设计与实现

案例 3 中介绍了教师功能中查询所授课程的成绩信息、其他类型的查询功能、增加学生成绩信息、修改学生成绩信息和删除学生成绩信息等功能的设计与实现。下面将介绍用户密码的修改。用户输入正确的用户名和密码后，弹出登录成功的消息框，然后不是直接跳转到查询页面，而是跳转到一个主页面，该页面用户可以选择进行密码修改还是进行成绩查询操作，如图 3-76 所示。用户单击密码修改则跳转到 updatepass.jsp 页面，运行效果如图 3-77 所示。

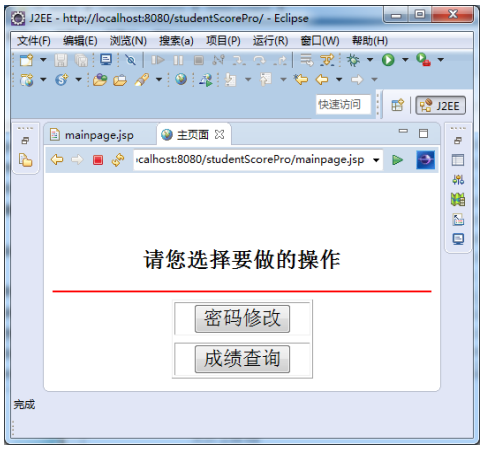


图 3-76　用户登录后的主页面

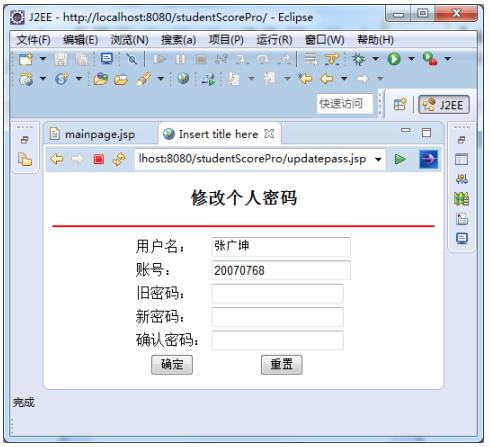


图 3-77　修改用户密码页面

用户输入旧密码、新密码和确认密码后，单击“确定”按钮则跳转到密码修改处理页面 teacherpassupdate.jsp，如果输入的旧密码正确，并且新密码和确认密码相等，则单击修改密码的“确定”按钮后则会弹出密码修改成功消息框。如图 3-78 所示。

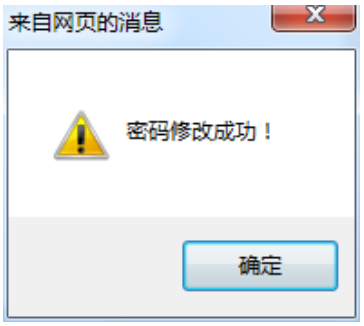


图 3-78　密码修改成功提示窗口

要实现以上功能需要在案例 3 的基础上进行下面代码修改：

把 checklogin.jsp 页面中的

```
   out.print("<script>  alert(' 登 录 成 功 ！ ');  location.href='teacher.jsp'
</script>");
```

修改为：

```
   out.print("<script>  alert(' 登 录 成 功 ！ ');  location.href='mainpage.jsp
'</script>");
```

然后按照建立 JSP 文件的方法，建立 mainpage.jsp 页面，该页面代码如下：

```
   <%@ page language="Java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>主页面</title>
</head>
<body>
    <br><br>
    <h2 align="center">请您选择要做的操作</h2>
    <hr color="red">
    <form>
        <table align="center" border="1" width=160>
            <tr>
                <td align="center"><input type="button" value="密
码修改"
                    style='font-size: 22px'
            onclick="window.location.href('updatepass.jsp')" /></td>
            </tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr>
                <td align="center"><input type="button" value="成
绩查询"
                    style='font-size: 22px'

    onclick="window.location.href('teacher.jsp')" /></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

然后再建立 updatepass.jsp 页面，该页面代码如下：

```
<%@page import="Java.sql.SQLException"%>
<%@page import="Java.sql.ResultSet"%>
<%@page import="Java.sql.Statement"%>
<%@page import="Java.sql.Connection"%>
<%@page import="Java.sql.DriverManager"%>
<%@ page language="Java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<% request.setCharacterEncoding("UTF-8");
response.setCharacterEncoding("UTF-8");
```

```
   String uid=session.getAttribute("uid").toString().trim();
   String upass=session.getAttribute("upass").toString().trim();
   Connection conn = null;
   Statement stmt = null;
   ResultSet rs = null;
   String tname=null;
   try {
      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
   conn   =   DriverManager.getConnection("jdbc:sqlserver://localhost:1433;
DatabaseName=ssms", "sa", "hh123456");
      stmt = conn.createStatement();
      String sql = "select tName from teacher where  tid='" + uid + "'";
      rs = stmt.executeQuery(sql);
      while(rs.next()){tname=rs.getString(1);
      }
   }
   catch( SQLException e){out.print("<script> alert('数据库操作有误！  ');
</script>");}
   finally{if(rs!=null)
   rs.close();
   if(stmt!=null)
   stmt.close();
   if(conn!=null)
   conn.close();}
   %>
      <form  name="updatepass" method="post" action="teacherpassupdate.jsp">
            <h3 align="center">修改个人密码</h3>
            <hr color="red">
            <table align="center">
                  <tr>
                        <td>用户名：</td>
            <td><input type="text" value="<%=tname%>" width="200" readonly/>
</td>
                  </tr>
                  <tr>
                        <td>账号：</td>
      <td><input type="text" value="<%=uid%>" width="200" readonly></td>
                  </tr>
                  <tr>
                        <td>旧密码：</td>
   <td><input type="password" name="oldpass" width="200"></td>
                  </tr>
                  <tr>
                        <td>新密码：</td>
      <td><input type="password" name="newpass" width="200"></td>
                  </tr>
                  <tr>
                        <td>确认密码：</td>
   <td><input type="password" name="surepass" width="200"></td>
                  </tr>
```

```
                    <tr align="center">
                          <td ><input type="submit" value="确定" /></td>
                          <td ><input type="reset" value="重置" /></td>
                    </tr>
              </table>
       </form>
   </body>
   </html>
```

最后再建立 teacherpassupdate.jsp 页面，该页面代码如下：

```
   <%@page import="Java.sql.SQLException"%>
   <%@page import="Java.sql.Statement"%>
   <%@page import="Java.sql.Connection"%>
   <%@page import="Java.sql.ResultSet"%>
   <%@page import="Java.sql.DriverManager"%>
   <%@page import="Java.sql.PreparedStatement"%>
   <%@ page language="Java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.
w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>修改用户密码</title>
   </head>
   <body>
      <%     request.setCharacterEncoding("UTF-8");
             response.setCharacterEncoding("utf-8");
             PreparedStatement ps = null;
             Connection conn = null;
             Statement st = null;
             String tid = session.getAttribute("uid").toString().trim();
             String tpass = session.getAttribute("upass").toString().trim();
             String oldpass = null, newpass = null, surepass = null;
             if (request.getParameter("oldpass") != null)
                    oldpass  =  new  String(request.getParameter("oldpass")).
trim();
             if (request.getParameter("newpass") != null)
                    newpass  =  new  String(request.getParameter("newpass")).
trim();
             if (request.getParameter("surepass") != null)
                    surepass = new String(request.getParameter("surepass")).
trim();
             if (tpass.equals(oldpass)) {
                    if (newpass.equals(surepass)) {
                           try {

      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

```
   onn    =    DriverManager.getConnection("jdbc:sqlserver://localhost:1433;
DatabaseName=ssms", "sa","hh123456");
              String sql = "update teacher set tPassword=? where tid=?";
                                    ps = conn.prepareStatement(sql);
                                    ps.setString(1, newpass);
                                    ps.setString(2, tid);
                                    int row1 = ps.executeUpdate();
                                    if (row1 != 0 )
                                    {   session.setAttribute("upass", newpass);
   out.print("<script>alert(' 密 码 修 改 成 功 !   ');location.href='login.jsp'
</script>");
                                    }else
   out.print("<script>alert('密码修改失败 !  ');location.href='mainpage.jsp'
</script>");
                             } catch (SQLException e) {
                             System.out.print("数据库操作异常 ! " + e);
                             } finally {
                                    conn.commit();
                                    if (ps != null)
                                           ps.close();
                                    if (conn != null)
                                           conn.close();
                             }
                      } else
   out.print("<script>alert('两次输入的密码不一致 请重新输入! ');location.href=
'updatepass.jsp'</script>");
              } else
       out.print("<script>alert('输入的原始密码不正确请重新输入!');location.href=
'updatepass.jsp'</script>");
       %>
       <br>
       <a href="teacher.jsp">返回</a>
   </body>
   </html>
```

习题

在本案例 3 的基础之上，实现学生用户登录后修改个人密码的功能。

案例 5　学生成绩管理系统的备份与恢复

通过前四个案例完成了教师功能模块的设计与实现，一个数据库系统在运行过程中，难免会出现意外情况，为了避免系统出现意外造成数据库中数据的丢失，数据库管理员要制定合理的备份方案，在系统出现故障时，通过备份文件进行数据库中数据的恢复，保证数据库存储的数据不会丢失。下面介绍以管理员身份登录后对系统数据进行备份和恢复操作。

任务 1　数据库备份

以管理员身份登录系统，如图 3-79 所示，输入管理原账号和密码，单击“登录”按钮，进行密码验证，验证通过后会弹出“登录成功”的消息提示框，密码错误会有“错误提示信息”，单击“确定”按钮后会跳转到主页面如图 3-76 所示。单击图 3-76 页面的“成绩查询”按钮，会跳转到管理员管理页面如图 3-80 所示：

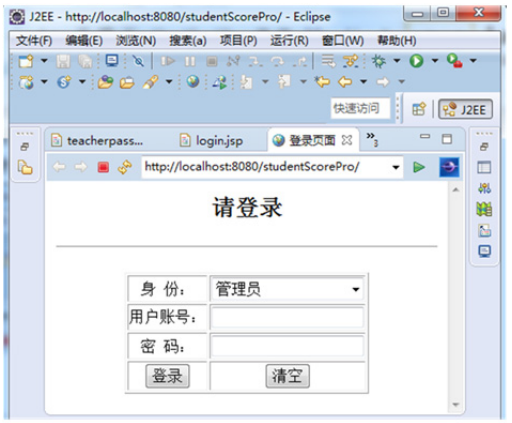


图 3-79　管理员身份登录

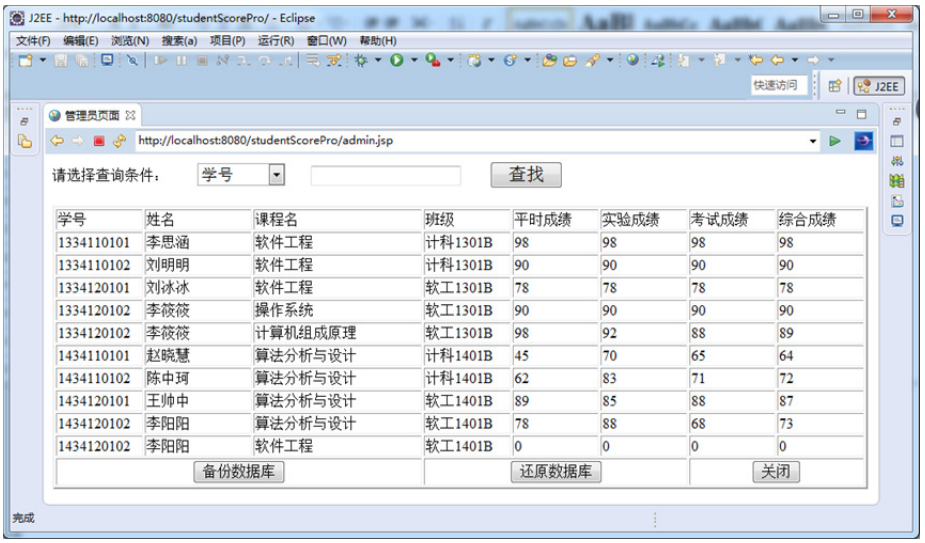

J2EE - http://localhost:8080/studentScorePro/ - Eclipse
文件(F) 编辑(E) 浏览(N) 搜索(a) 项目(P) 运行(R) 窗口(W) 帮助(H)
快速访问
J2EE
管理员页面
http://localhost:8080/studentScorePro/admin.jsp
请选择查询条件：
学号
查找
学号 姓名 课程名 班级 平时成绩 实验成绩 考试成绩 综合成绩
1334110101 李思涵 软件工程 计科1301B 98 98 98 98
1334110102 刘明明 软件工程 计科1301B 90 90 90 90
1334120101 刘冰冰 软件工程 软工1301B 78 78 78 78
1334120102 李筱筱 操作系统 软工1301B 90 90 90 90
1334120102 李筱筱 计算机组成原理 软工1301B 98 92 88 89
1434110101 赵晓慧 算法分析与设计 计科1401B 45 70 65 64
1434110102 陈中珂 算法分析与设计 计科1401B 62 83 71 72
1434120101 王帅中 算法分析与设计 软工1401B 89 85 88 87
1434120102 李阳阳 算法分析与设计 软工1401B 78 88 68 73
1434120102 李阳阳 软件工程 软工1401B 0 0 0 0
备份数据库
还原数据库
关闭
完成

图 3-80　管理员页面

要出现上述页面需要在 login.jsp 中进行如下代码修改：

首先在

```
   <tr>
            <td align="center">用户账号：</td>
```

代码前面加入如下代码：

```
   <tr><td align="center">身  份：</td>
                  <td><select name="logintype" style="width:160px;font-size:
14px">
                   <option value="admin">管理员</option>
                   <option value="teacher">老师</option>
                   </select></td>
                   </tr>
```

然后在 checklogin.jsp 页面找到如下代码：

```
   String userid = request.getParameter("userid").trim();加上如下代码：
   if(request.getParameter("logintype")!=null)
             logintype= request.getParameter("logintype").trim();
             session.setAttribute("logintype", logintype);
```

把其中的代码

```
   String sql = "select * from teacher where tid= '"+userid+"'  and  tPassword="+
upass;
```

修改为

```
   String sql = null;
                    if (logintype.equals("admin"))
   sql = "select * from administrator where aid='" + uid + "' and aPassword='"
+ upass + "'";
                    else if (logintype.equals("teacher"))
   sql = "select * from teacher where tid= '" + uid + "' and  tPassword= '" +
upass + "'";
```

再把其中的代码

```
   out.print("<script>  alert(' 登 录 成 功 ！ ');  location.href='teacher.jsp'
</script>");/
```

修改为：

```
   out.print("<script>  alert('登录成功！ ');  location.href='mainpage.jsp'关
</script>");
```

然后建立 JSP 页面 mainpage.jsp 页面，其代码如下：

```
   <%@ page language="Java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>主页面</title>
   </head>
   <body>
      <br><br><br>
      <h2 align="center">请您选择要做的操作</h2>
      <hr color="red">
      <form>
            <%     String logintype = "";
                   logintype = session.getAttribute("logintype").toString();
            %>
```

```
            <table align="center" border="1" width=160>
                  <tr>
                        <td align="center"><input type="button" value="密
码修改"
                              style='font-size: 22px'
      onclick="window.location.href('updatepass.jsp')" /></td>
                  </tr>
                  <tr></tr>    <tr></tr>    <tr></tr>
                  <tr>
                        <%
                              if (logintype.equals("teacher")) {
                        %>
                        <td align="center"><input type="button" value="成
绩查询"
                              style='font-size: 22px'

      onclick="window.location.href('teacher.jsp')" /></td>
                        <%
                              } else if (logintype.equals("admin")) {
                        %>
   <td align="center"><input type="button" value="成绩查询"
   style='font-size:  22px'  onclick="window.location.href('admin.jsp')"  />
</td>
                        <%
                              }
                              if (logintype.equals("student")) {
                        %>
                        <td align="center"><input type="button" value="成
绩查询"
      style='font-size: 22px'     onclick="window.location.href('student.jsp')"
/></td>
                        <%
                              }
                        %>
                  </tr>
            </table>
      </form>
   </body>
   </html>
```

然后再建立一个 JSP 页面 admin.jsp，该页面代码如下：

```
   <%@ page language="Java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <%@ page import="Java.util.*"%>
   <%@ page import="Java.sql.*"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
   <title>管理员页面</title>
   </head>
   <body>
      <script language="Javascript">
            function change() {
                  var option = document.getElementById("select").options;
                  var      scoreoption      =      document.getElementById
("selectscoretype"). options;
                  for (var i = 0; i < option.length; i++)
                        option[i].value = option[i].innerHTML;
                        for (var j = 0; j < scoreoption.length; j++)
                        scoreoption[j].value = scoreoption[j].innerHTML;
                  document.admin.submit();
            }
            function changeItem() {
                  var selectitem = admin.select.value;
                  window.location.href = "admin.jsp?select=" + selectitem;
            }
            function scorechangeItem() {
                  var selectitem = admin.select.value;
                  var selectscoreitem = admin.selectscoretype.value;
                  window.location.href = "admin.jsp?select=" + selectitem
                              + "&selectscoretype=" + selectscoreitem;
            }
      </script>
      <%     request.setCharacterEncoding("utf-8");
            response.setCharacterEncoding("utf-8");
            String selectitem = "学号", selectscoreitem = "精确查找";
            if (request.getParameter("select") != null)
                  selectitem  =  request.getParameter("select").toString().
trim();
            if (request.getParameter("selectscoretype") != null)
   selectscoreitem = request.getParameter("selectscoretype").toString().trim();
      %>
      <form action=teacher.jsp name="teacher" method="post">
            请选择查询条件：           <select
                  style='font-size:    18px'    name="select"    onChange=
"changeItem()">
   <option value="学号" <%if (selectitem.equals("学号")) {%> selected <%}%>>
学号</option>
   <option value="姓名" <%if (selectitem.equals("姓名")) {%> selected <%}%>>
姓名</option>
            <option value="综合成绩" <%if (selectitem.equals("综合成绩")) {%>
selected
                        <%}%>>综合成绩</option>
            </select>      
            <%    if (selectitem.equals("学号") || selectitem.equals("姓名"))
{
            %>
            <input type="text" name="searchcontent" style="width: 160px" />
```

```
            <%     } else if (selectitem.equals("综合成绩")) {
            %>请选择成绩查找方式：    <select style='font-size: 18px'
                  name="selectscoretype" onChange="scorechangeItem()">
            <option value="精确查找" <%if (selectscoreitem.equals("精确查找"))
{%>
                        selected <%}%>>精确查找</option>
            <option value="比较查找" <%if (selectscoreitem.equals("比较查找"))
{%>
                        selected <%}%>>比较查找</option>
            <option value="范围查找" <%if (selectscoreitem.equals("范围查找"))
{%>
                        selected <%}%>>范围查找</option>
            <option value="集合查找" <%if (selectscoreitem.equals("集合查找"))
{%>
                        selected <%}%>>集合查找</option>
            <option value="无成绩查找" <%if (selectscoreitem.equals("无成绩查
找")) {%>
                        selected <%}%>>无成绩查找</option>
            </select>      
            <%
                  }
      if ((selectitem.equals("综合成绩")) && ((selectscoreitem.equals("精确查
找")) || (selectscoreitem.equals("比较查找"))
                              || (selectscoreitem.equals("集合查找")))) {
      %>
   请输入查找的成绩条件： <input type="text" name="searchscorecontent" width=200
                  style="width: 40px" />
            <%     }
      if ((selectitem.equals("综合成绩")) && (selectscoreitem.equals("范围查找
"))) {
            %>
            查找成绩开始值<input type="text" name="beginscore" width=200
                  style="width: 40px" />     成绩
终止值<input  type="text" name="endscore" width=200 style="width: 40px" />
            <%
                  }
            %>
                   <input type="submit" value=
"查找"
                  style='font-size: 20px; width: 80px; height: 30px;' /><br>
<br>
            <table border="2" width="900">
                  <tr>   <td width="80">学号</td>
                        <td width="100">姓名</td>
                        <td width="160">课程名</td>
                        <td width="80">班级</td>
                        <td width="80">平时成绩</td>
                        <td width="80">实验成绩</td>
                        <td width="80">考试成绩</td>
                        <td width="80">综合成绩</td>
```

```
                    </tr>
                    <%     request.setCharacterEncoding("UTF-8");
                           response.setCharacterEncoding("utf-8");
                           String    tid    =    session.getAttribute("uid").
toString().trim();
                           String   tpass   =   session.getAttribute("upass").
toString().trim();
                           Connection conn = null;
                           Statement stmt = null;
                           ResultSet rs = null;
                           try {  String selectcontent = null, select = null;
                                  if
(request.getParameter("searchcontent") != null)
              selectcontent = request.getParameter("searchcontent"). toString().
trim();
                                  if (request.getParameter("select") != null)
       selectitem = request.getParameter("select").toString().trim();
    conn           =           DriverManager.getConnection("jdbc:sqlserver://
localhost:1433;DatabaseName=ssms", tid, tpass);
                                  stmt = conn.createStatement();
                                  String sql = "select  *  from  v_stu_score";
                                  if (request.getParameter("select") != null)
{
       if (selectitem.equals("学号") || selectitem.equals("姓名")) {
    if (!("".equals(selectcontent) || selectcontent == null)) {
       if (selectitem.equals("学号"))
       sql += "and student.sid='" + selectcontent + "'";
       else if (selectitem.equals("姓名"))
       sql += "and sName like '%" + selectcontent + "%'";
                                                }
                                          } else if (selectitem.equals("综合成
绩")) {
                                                String  searchscorecontent  =
null;
                                                int beginscore = 0, endscore =
0;
    if (request.getParameter("searchscorecontent") != null)
    searchscorecontent     =     request.getParameter("searchscorecontent").
toString().trim();
    if (!("".equals(searchscorecontent) || searchscorecontent == null)) { if
(selectscoreitem.equals("精确查找"))
    sql += "and totalScore=" + searchscorecontent;
       else if (selectscoreitem.equals("比较查找"))
       sql += "and totalScore" + searchscorecontent;
       else if (selectscoreitem.equals("集合查找"))
       sql += "and totalScore in (" + searchscorecontent + ")";
       } else if (selectscoreitem.equals("无成绩查找"))
       sql += "and totalScore is null";
    else if (selectscoreitem.equals("范围查找")) {
    if (request.getParameter("beginscore") != null
```

```
       && !("".equals(request.getParameter("beginscore").toString().trim()))
)
   beginscore = Integer.valueOf((request.getParameter("beginscore"). toString().
trim()));
   if (request.getParameter("endscore") != null

       && !("".equals(request.getParameter("endscore").toString().trim())))
   endscore  =  Integer.valueOf(request.getParameter("endscore").toString().
trim());
   if (beginscore != 0 && endscore != 0)
   sql += "and totalScore  between " + beginscore + " and " + endscore;
                                                  }
                                              }
                                  }
                                  rs = stmt.executeQuery(sql);
       if (selectscoreitem.equals("无成绩查找") ) {
   String usualResults, examResults , experimentalResults , totalScore;
             while (rs.next()) {
                   if (rs.getInt(5) == 0) usualResults= "";
                   else  usualResults= String.valueOf(rs.getInt(5));
             if (rs.getInt(6) == 0) examResults = "";
             else   examResults= String.valueOf(rs.getInt(6));
             if (rs.getInt(7) == 0) experimentalResults= "";
             else   experimentalResults= String.valueOf(rs.getInt(7));
             if (rs.getInt(8) == 0) totalScore = "";
             else   totalScore= String.valueOf(rs.getInt(8));%>
                   <tr>   <td><%=rs.getString(1)%></td>
                          <td><%=rs.getString(2)%></td>
                          <td><%=rs.getString(3)%></td>
                          <td><%=rs.getString(4)%></td>
                          <td><%=usualResults%></td>
                          <td><%=examResults%></td>
                          <td><%=experimentalResults%></td>
                          <td><%=totalScore%></td>
                   </tr>
                   <%
                   }} else {
                   while  (rs.next()) {
                   %>
                   <tr>   <td><%=rs.getString(1)%></td>
                          <td><%=rs.getString(2)%></td>
                          <td><%=rs.getString(3)%></td>
                          <td><%=rs.getString(4)%></td>
                          <td><%=rs.getInt(5)%></td>
                          <td><%=rs.getInt(6)%></td>
                          <td><%=rs.getInt(7)%></td>
                          <td><%=rs.getInt(8)%></td>
                   </tr>
                   <%
                   }
```

```
                        }
                        } catch (SQLException e) {
            out.print("<script> alert('数据库操作出现异常！'+e);</script>");
                        } finally {
                              if (rs != null)
                                    rs.close();
                              if (stmt != null)
                                    stmt.close();
                              if (conn != null)
                                    conn.close();
                        }
                  %>
                  <tr>
      <td colspan="3" align="center"><input type="button" value="备份数据库"
      style='font-size:  16px'  onclick="window.location.href('backupdata.
jsp')" /></td>
   <td colspan="3" align="center"><input type="button" value="还原数据库"
      style='font-size:  16px'  onclick="window.location.href('restoredata.
jsp')" /></td>
   <td colspan="2" align="center"><input id="close" type="button"
                              style='font-size: 16px' value="关闭" name=
"close"
                              onclick="window.close()"></td>
                  </tr>
            </table>
      </form>
   </body>
   </html>
```

在管理员操作页面具有教师操作页面的所有数据查询功能，同时具有数据的备份和恢复功能。单击其中的“备份数据库”按钮会弹出如图 3-81 所示备份数据库界面，要实现图 3-81 所示页面效果需要建立 JSP 文件 backupdata.jsp 页面，该页面代码如下：

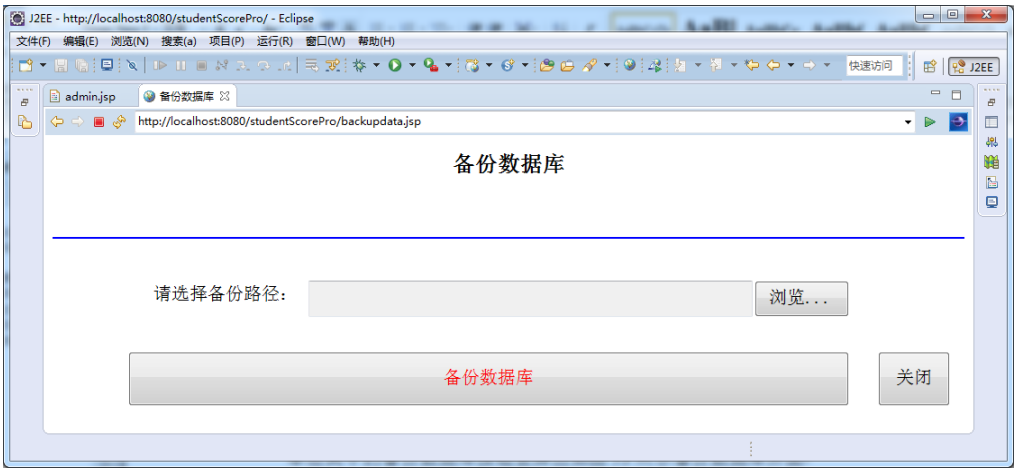


图 3-81　备份数据库窗口

```
   <%@page import="Java.util.Date"%>
   <%@page import="Java.sql.SQLException"%>
   <%@page import="Java.sql.ResultSet"%>
   <%@page import="Java.sql.Statement"%>
   <%@page import="Java.sql.Connection"%>
   <%@page import="Java.sql.DriverManager"%>
   <%@ page language="Java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>备份数据库</title>
   </head>
   <body>
      <script language='Javascript'>
            function getFullPath(obj) {
                  if (obj) {
                        if (window.navigator.userAgent.indexOf("MSIE") >=
1) //ie
                        {     obj.select();
                              return
document.selection.createRange().text;
                   else if (window.navigator.userAgent.indexOf("Firefox") >=
1) //firefox
                        {     if (obj.files) {
                                    return
obj.files.item(0).getAsDataURL();
                              }
                              return obj.value;
                        }
                        return obj.value;
                  }
            }
      </script>
      <form name="backup" action="backup.jsp" method="post">
            <h2 align="center">备份数据库</h2>
            <br> <br>
            <hr color="blue">
            <br> <br>
               
            
 
            <span style="font-size: 20px;">请选择备份路径: </span>  
<input       type="file" style='font-size: 20px; width: 600px; height: 40px;'

      onchange="document.getElementById('filepath').value=getFullPath(this)
;" />
                 <input type="hidden" id="filepath"
```

```
                    name="filepath" /> <br>
            <br> <br>
                   
        
            <input type="submit" value="备份数据库" size="600"
                    style='font-size: 20px; width: 800px; height: 60px; color:
red;' />
                    <input  id="close"  type=
"button"
                    style='font-size: 20px; height: 60px; width: 80px;' value=
"关闭"
                    name="close" onclick="window.close()">
      </form>
   </body>
   </html>
```

在图 3-81 中首先单击其中的“浏览”按钮选择备份路径，单击“浏览”按钮会弹出如图 3-82 所示的“选择备份文件”的窗口，选择事先建立好的备份文件，然后单击“打开”按钮，会返回图 3-81 窗口，这时选择的备份文件会在备份路径中显示出来，如图 3-83 所示。

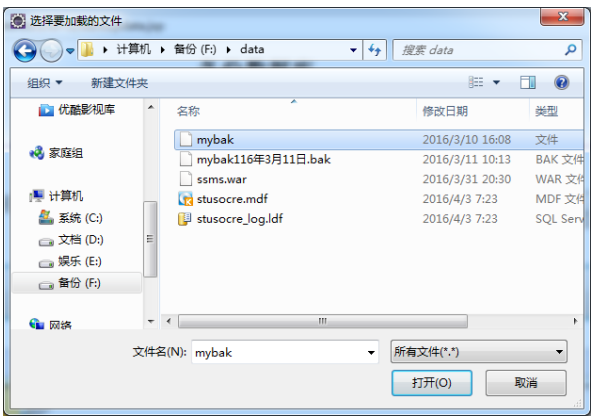


图 3-82　选择备份文件

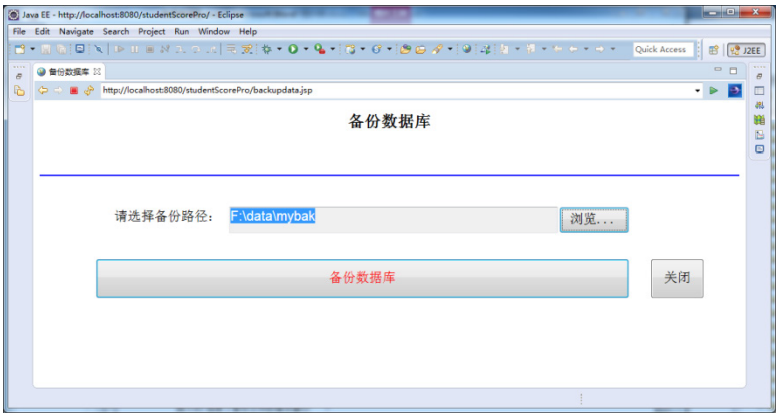


图 3-83　选择了备份文件的备份窗口

然后单击“备份数据库”按钮，备份数据库需要一定时间，完成备份后会弹出如图 3-84 所示的“备份成功”消息提示窗口，完成了数据库的备份工作。要实现上述的数据库备份功能，需要建立 JSP 文件 backup.jsp，其代码如下：

图 3-84　成功提示窗口

```
   <%@page import="Java.util.Date"%>
   <%@page import="Java.sql.ResultSet"%>
   <%@page import="Java.sql.Statement"%>
   <%@page import="Java.sql.Connection"%>
   <%@page import="Java.sql.DriverManager"%>
   <%@page import="Java.sql.SQLException"%>
   <%@ page language="Java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>备份数据库</title>
   </head>
   <body>
   <%
            request.setCharacterEncoding("UTF-8");
            response.setCharacterEncoding("UTF-8");
            String aid = session.getAttribute("uid").toString().trim();
            String apass = session.getAttribute("upass").toString().trim();
            Connection conn = null;
            Statement stmt = null;
            ResultSet rs = null;
            try {

      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
   conn   =   DriverManager.getConnection("jdbc:sqlserver://localhost:1433;
DatabaseName=ssms", aid, apass);
                  stmt = conn.createStatement();
                  Date today = new Date();
   String date = (today.getYear()) + "年" + (today.getMonth() + 1) + "月" +
(today.getDate()) + "日";
                  String p = request.getParameter("filepath");
                  if (request.getParameter("filepath") != null) {
   String path = request.getParameter("filepath").toString().trim()+date +
".bak";
            String sql;
            sql = "backup  database ssms to disk= '" + path + "' with init";
            stmt.executeUpdate(sql);
```

```
     out.print("<script> alert('备份 成功！ ');location.href='backupdata.jsp'
</script>");
                    } else { out.print("<script> alert('请选择备份路径!!  ');
</script>");
                    }
              } catch (Exception e) {
                    out.print("<script>  alert('数据库操作出现异常！ '+e);
</script>");
              } finally {
                    if (rs != null)
                          rs.close();
                    if (stmt != null)
                          stmt.close();
                    if (conn != null)
                          conn.close();
              }
       %>
   </body>
   </html>
```

习题

认真思考本案例任务 1，在案例 4 的基础之上，实现管理员用户登录后进行数据库的备份操作。

任务 2　数据库恢复

备份数据库是为了在系统出现故障时，通过备份文件进行数据库的恢复（还原）工作，避免数据的丢失，下面介绍数据库恢复（还原）功能的设计与实现。

在图 3-80 中的管理员操作页面单击其中的“还原数据库”会弹出如图 3-85 所示的“恢复数据库”窗口。

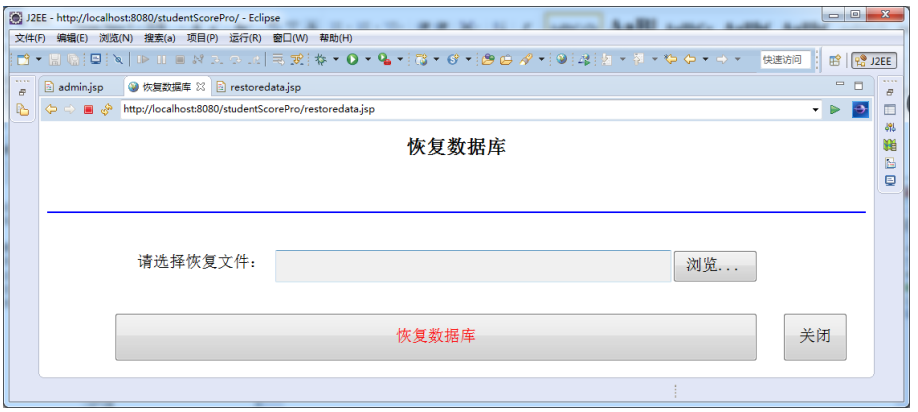


图 3-85　数据库恢复窗口

要实现图 3-85 所示的恢复界面，需要建立名为 restoredata.jsp 的文件，该文件的代码如下：

```
   <%@page import="Java.util.Date"%>
   <%@page import="Java.sql.SQLException"%>
   <%@page import="Java.sql.ResultSet"%>
   <%@page import="Java.sql.Statement"%>
   <%@page import="Java.sql.Connection"%>
   <%@page import="Java.sql.DriverManager"%>
   <%@ page language="Java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>恢复数据库</title>
   </head>
   <body>
      <script language='Javascript'>
            function getFullPath(obj) {
                  if (obj) {
                        if (window.navigator.userAgent.indexOf("MSIE") >=
1) //ie
                        {     obj.select();
                              return
document.selection.createRange().text;
            } else if (window.navigator.userAgent.indexOf("Firefox") >= 1)
//firefox
                        {           if (obj.files) {
                              return
obj.files.item(0).getAsDataURL();
                              }
                              return obj.value;
                        }
                        return obj.value;
                  }
            }
      </script>
      <form name="restore" action="restore.jsp" method="post">
            <h2 align="center">恢复数据库</h2>
            <br> <br>
            <hr color="blue">
            <br> <br>
               
            
 
            <span style="font-size: 20px;">请选择恢复文件：</span>  
<input type="file" style='font-size: 20px; width: 600px; height: 40px;'

      onchange="document.getElementById('filepath').value=getFullPath(this);"
/>
                  <input  type="hidden"  id="filepath"
```

```
        name="filepath" />
            <br><br> <br>
                    
        
                <input type="submit" value="恢复数据库" size="600"
                        style='font-size: 20px; width: 800px; height: 60px; color:
red;' />
                        <input  id="close"  type=
"button"
                        style='font-size: 20px; height: 60px; width: 80px;' value=
"关闭"
                        name="close" onclick="window.close()">
        </form>
    </body>
    </html>
```

首先单击其中的“浏览”按钮选择备份文件，会弹出如图 3-86 所示的“选择要加载的文件”对话框，选择了还原文件后单击“打开”按钮，又返回到图 3-85 所示的页面，这时选择的还原文件已经显示的还原文件路径中如图 3-87 所示，单击“还原数据库”命令按钮，完成数据库的还原操作。成功还原数据库后会弹出一个如图 3-88 所示的“数据库成功恢复”的消息窗口。

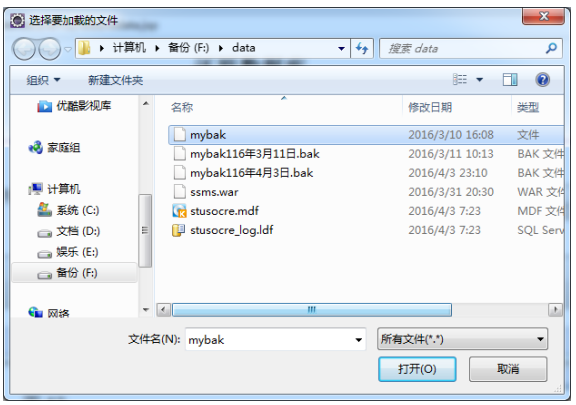


图 3-86　选择还原文件

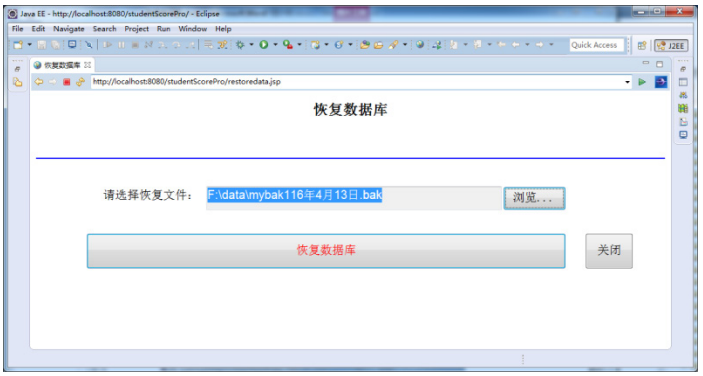


图 3-87　恢复数据库窗口

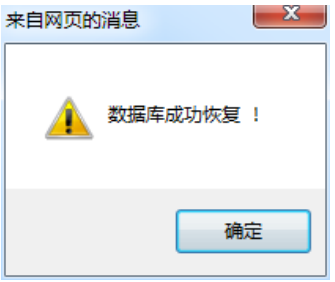


图 3-88　成功提示窗口

要实现如上功能，需要建立名为 restore.jsp 的文件，该文件代码如下：

```
    <%@page import="Java.sql.ResultSet"%>
    <%@page import="Java.sql.Statement"%>
    <%@page import="Java.sql.Connection"%>
    <%@page import="Java.sql.DriverManager"%>
    <%@page import="Java.sql.SQLException"%>
    <%@ page language="Java" contentType="text/html; charset=UTF-8"
       pageEncoding="UTF-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>恢复数据库</title>
    </head>
    <body>
    <%          request.setCharacterEncoding("UTF-8");
                response.setCharacterEncoding("UTF-8");
                String aid = session.getAttribute("uid").toString().trim();
                String apass = session.getAttribute("upass").toString().trim();
                Connection conn = null;
                Statement stmt = null;
                ResultSet rs = null;
                try {  Class.forName("com.microsoft.sqlserver.jdbc.
SQLServerDriver");
    conn    =    DriverManager.getConnection("jdbc:sqlserver://localhost:1433;
DatabaseName=ssms", aid, apass);
                      stmt = conn.createStatement();
                            String p = request.getParameter("filepath");
                      if (request.getParameter("filepath") != null) {
                            String  path  =  request.getParameter("filepath").
toString().trim();
       String sql= "alter database ssms set offline with rollback immediate;use
master;"+
                            "restore database ssms from disk='"+path+"' with
replace;alter database ssms set online with rollback immediate";
                         stmt.executeUpdate(sql);
              out.print("<script> alert('数据库成功恢复 ！ ');location.href=
```

```
'restoredata.jsp' </script>");          } else {      out.print("<script>
alert('请选择恢复路径!!  ');</script>");
                    }
              } catch (Exception e) {
                    out.print("<script>  alert('数据库操作出现异常！ '+e);
</script>");
              } finally {
                    if (rs != null)
                          rs.close();
                    if (stmt != null)
                          stmt.close();
                    if (conn != null)
                          conn.close();
              }
       %>
   </body>
   </html>
```

习题

认真思考本案例的任务 2，在本案例任务 1 的基础之上，实现管理员用户登录后进行数据库的恢复操作。

相关知识

一、数据安全性

随着计算机网络的飞速发展，信息安全问题日趋明显，保证信息安全的一个重要方法是数据备份。只要发生数据传输、数据存储和数据交换，均有可能产生数据故障。这时，如果没有采取数据备份和数据恢复手段与措施，就会导致数据的丢失，数据丢失有时造成的损失是无法弥补与估量的。

数据故障的形式是多种多样的。一般将数据故障划分为系统故障、事务故障和介质故障三大类。从信息安全数据库备份与恢复方案的角度出发，实际上第三方或敌方的“信息攻击”，也会产生不同种类的数据故障。例如：计算机病毒型、特洛伊木马型、“黑客”入侵型、逻辑炸弹型等，这些故障将会造成的后果有：数据丢失、数据被修改、增加无用数据及系统瘫痪等。作为系统管理员，要千方百计地维护系统和数据的完整性与准确性。通常采取的措施有：安装防火墙，防止“黑客”入侵；安装防病毒软件，采取存取控制措施；选用高可靠性的软件产品；增强计算机网络的安全性。威胁数据的安全，造成系统失效的主要原因有以下几个方面：

（1） 硬盘驱动器损坏；

（2） 人为错误；

（3） 黑客攻击；

（4） 病毒；

（5） 自然灾害；

（6） 电源浪涌；

（7） 磁干扰。

因此，数据备份与数据恢复是保护数据的最后手段，也是防止主动型信息攻击的最后一道防线。

二、SQL Server 数据库备份的方案

按照 SQL Server 数据库备份数据量的不同，将 SQL Server 数据库备份分为完整备份、差异备份、事务日志备份三种，通常根据数据安全性的要求，推荐的备份方式为每周一次完整备份，每天一次差异备份，每半个小时一次事务日志备份。按照 SQL Server 数据库备份过程不同分为手动备份和自动备份两种，由于目的不同，一般手动备份应用于数据库备份不常出现的环境，例如课堂教学环境中，而自动备份通常在实际项目维护中使用。默认情况下，为 sysadmin 固定服务器角色以及 db_owner 和 db_backupoperator 固定数据库角色的成员授予 backup database 和 backup log 权限。备份设备的物理文件的所有权和权限问题可能会妨碍备份操作，要实现数据库备份 SQL Server 要求必须能够读取和写入设备；运行 SQL Server 服务的账户必须具有写入权限。备份文件存放磁盘需要与数据库文件存放磁盘分开，避免磁盘 IO 冲突。备份执行时间与数据库作业执行时间错开，避免备份影响数据库作业的执行。

1. 数据库备份

1） 自动备份。

SQL Server 维护计划功能可以较好地实现自动备份，在使用该功能前启动数据库管理器上的 SQL Server 代理功能，如图 3-89 所示。

名称	状态	启动模式	登录身份为	进程 ID	服务类型
SQL Server Integration Services 10.0	正在运行	自动	LocalSystem	2288	
SQL Server (SQLEXPRESS)	正在运行	自动	NT AUTHORITY\...	2400	SQL Server
SQL Full-text Filter Daemon Launcher (MSSQLSERVER)	正在运行	手动	NT AUTHORITY\L...	3732	
SQL Server (MSSQLSERVER)	正在运行	自动	LocalSystem	2536	SQL Server
SQL Server Analysis Services (MSSQLSERVER)	正在运行	自动	NT AUTHORITY\...	2568	Analysis Server
SQL Server Reporting Services (MSSQLSERVER)	正在运行	自动	NT AUTHORITY\...	2784	Report Server
SQL Server 代理 (SQLEXPRESS)	已停止	其他(“启动”、“...	NT AUTHORITY\...	0	SQL Agent
SQL Server Browser	已停止	其他(“启动”、“...	NT AUTHORITY\L...	0	
SQL Server 代理 (MSSQLSERVER)	正在运行	手动	NT AUTHORITY\...	6076	SQL Agent

图 3-89 启用 SQL Server 代理（MSQLSERVER）

目标任务：使用 SQL Server 2008 自带的维护计划自动备份数据库，并且自动删除超过设定期限的历史备份文件：

① 每天 0 点自动备份指定学生成绩管理数据库 ssms，备份文件放在 E:\book 下。

② 备份文件保留 1 周，每天备份完数据库后自动删除超过 1 周的备份文件。

前提条件：

① 已经安装 SQL Server Integration Services 服务。

② 已经安装 SQL Server 代理（MSSQLSERVER）服务。若已经成功安装 SQL Server 代理服务，如图 3-90 所示。

如果未安装 SQL Server Integration Services 服务，依然可以创建维护计划，但是不会

生效。

（1） 完整备份。

① 选择对象资源管理器下的“管理”，对“维护计划”单击右键，在弹出的快捷菜单中双击“维护计划向导”，如图 3-91 所示。

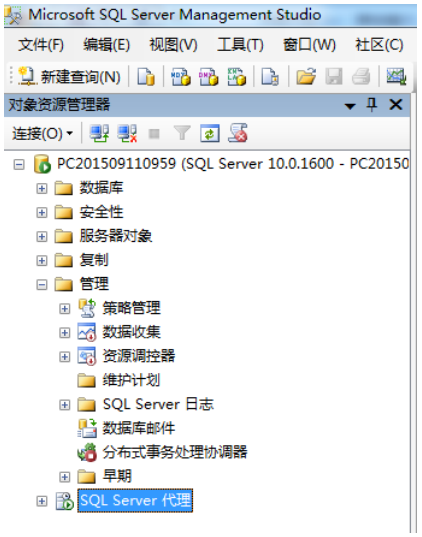


图 3-90　成功安装 SQL Server 代理

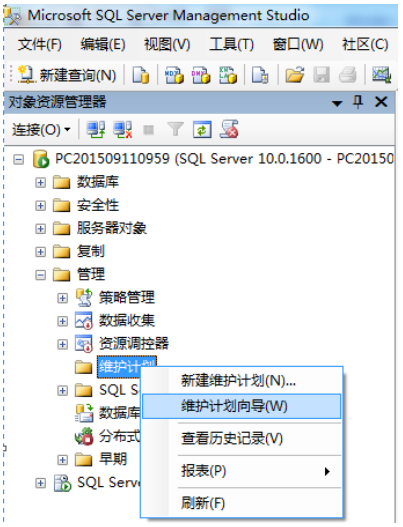


图 3-91　选择维护计划向导

双击“维护计划向导”会弹出 SQL Server 维护计划向导功能界面如图 3-92 所示，在“SQL Server 维护计划向导”功能界面中，单击“下一步”进入“选择计划属性”界面如图 3-93 所示。在“选择计划属性”界面中修改计划名称为“学生成绩管理数据库完整备份”，并给维护计划说明为“每周日晚上 12 点执行该数据库的完整备份”，如图 3-93 所示。

图 3-92　维护计划向导功能界面

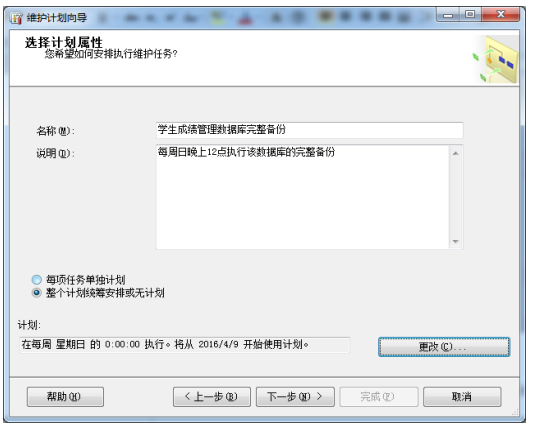


图 3-93　选择计划属性界面

② 由于备份数据库 ssms 和删除历史备份文件是两项任务，所以在“选择计划属性”界面中选择“整个计划统筹安排或无计划”，同时单击“更改”按钮，进入“时间计划设置”界面为维护计划指定执行周期，如图 3-94 所示。

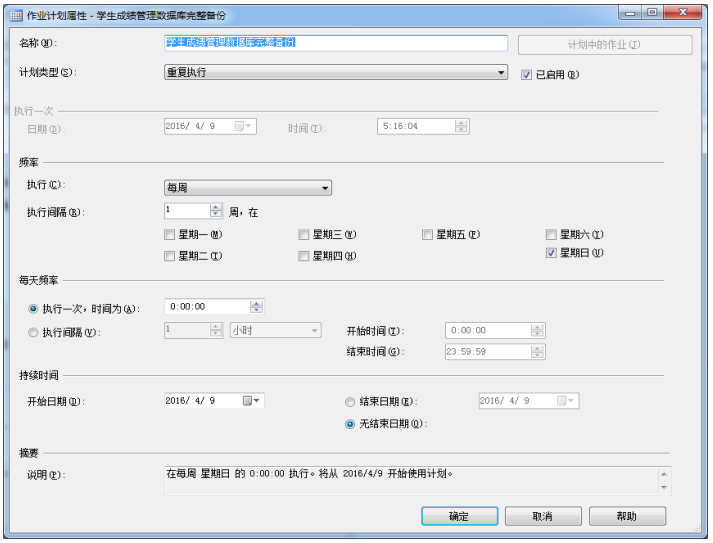


图 3-94　时间计划设置界面

在图 3-94 中计划类型选择“重复执行”，频率即备份周期选择“每周”，在每周日凌晨执行该计划，然后单击“确定”，返回“选择计划属性”界面后，再单击“下一步”按钮。进入如图 3-95 所示的备份数据库任务的“选择维护任务”界面。

③ 在该界面中选择“备份数据库（完整）”和“清除维护任务”两项维护任务。

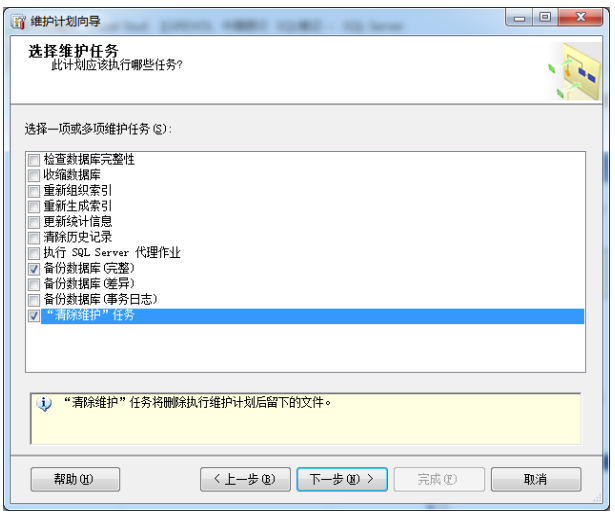


图 3-95　选择维护任务界面

在“选择维护任务”界面中单击“下一步”，进入如图 3-96 所示的“选择维护任务顺序”的界面，这里的顺序是先“备份数据库”，再执行“清除维护”任务，删除历史备份计划创建后，需要将过期的备份数据库自动删除，以保障磁盘空间的高效利用。在当前维护计划中建立清除维护任务，用于删除保留时间超过设定值的备份文件。在“选择维护任务顺序”的界面下，单击“下一步”。进入如图 3-97“定义‘备份数据库（完整）’任务”界

面。该界面设置数据库完备维护计划相关信息，包括备份的数据库、备份集过期时间、备份文件存储路径。单击“数据库”右侧的黑色箭头，在弹出的选项中选择需要备份的数据库为“ssms”后，单击“确定”，返回“定义‘备份数据（完整）’任务”界面，然后指定备份文件存放的目录为“E:\book”目录下。如图 3-98 所示。确定备份数据库及其他相关信息后，单击“下一步”。进入如图 3-99“定义清除维护任务”界面。

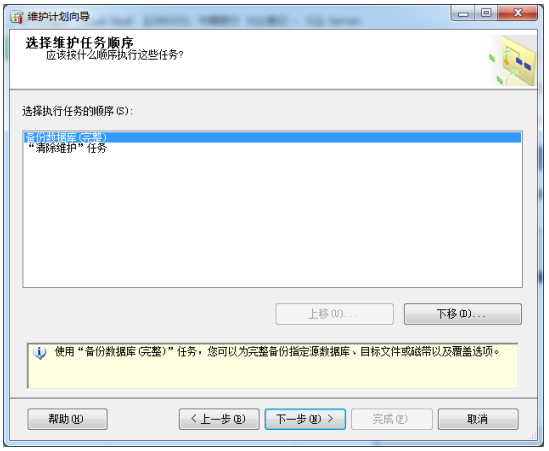


图 3-96　选择维护任务顺序界面

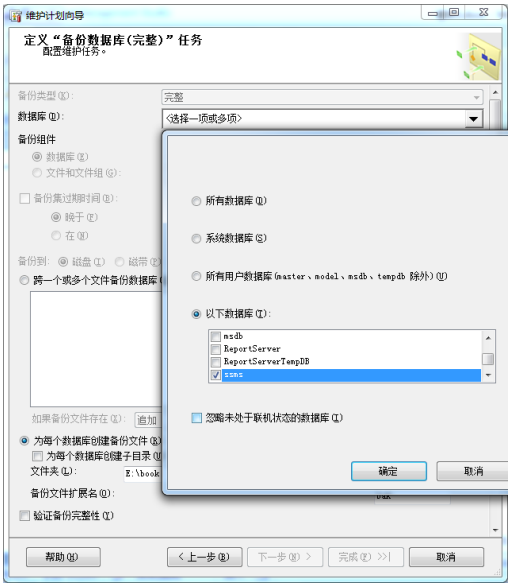


图 3-97　定义“备份数据（完整）”任务界面

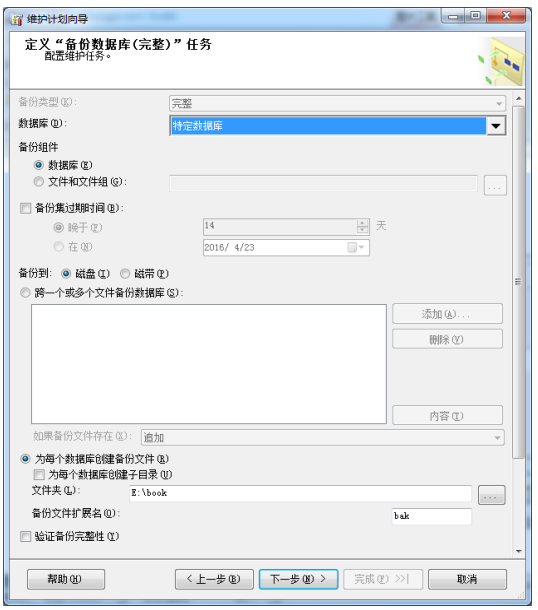


图 3-98　定义“备份数据（完整）”任务

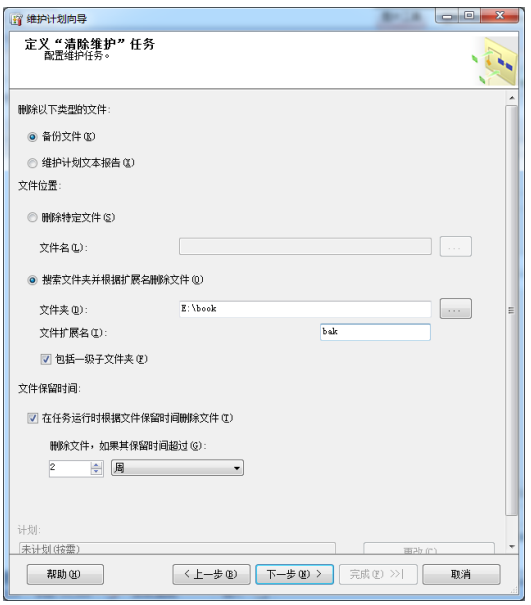


图 3-99　定义清除维护任务界面

④ 该界面需要指定删除文件类型及删除文件的位置等信息。默认删除文件的类型为

“备份文件”，指定文件的位置为“搜索文件夹并根据扩展名删除文件”，并且指定文件夹为E:\book，同时输入删除的文件扩展名为“bak”的数据库备份文件，文件保留时间为 2 周。特别注意：文件扩展名填“bak”，而不要填“.bak”，否则不会删除原来的备份文件。

⑤ 然后，单击“下一步”。进入如图 3-100 所示的“选择报告选项”界面，在该界面中，提示用户将报告写入文本文件的位置信息，确定后单击“下一步”，进入如图 3-101 所示“维护计划向导”的“完成该向导”界面，直接单击“完成”按钮，则数据库完整备份计划完成，如图 3-102 所示。

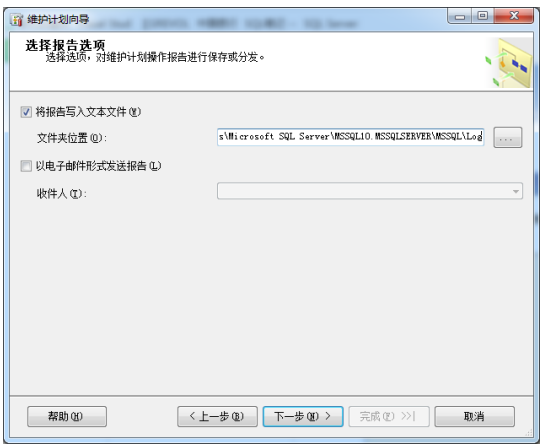


图 3-100　选择报告选项界面

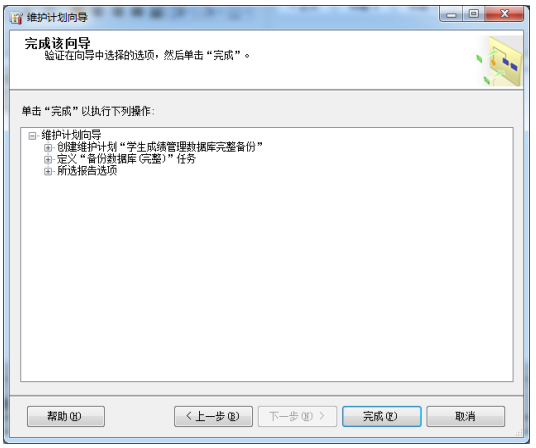


图 3-101　完成该向导界面

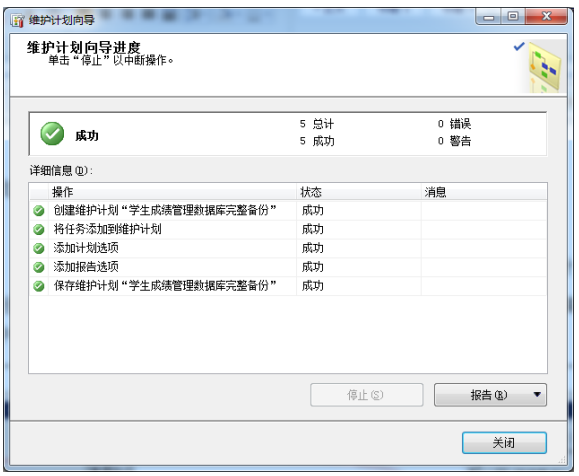


图 3-102　数据库完整备份计划完成界面

⑥ 刷新“对象资源管理器”下的“维护计划”文件夹，可以看到创建的维护计划如图 3-103 所示。

⑦ 双击新建的“学生成绩管理数据库完整备份”维护计划或单击右键选择“修改”，可以看到维护计划的内容，并修改，修改后点击保存即可，如图 3-104 所示。

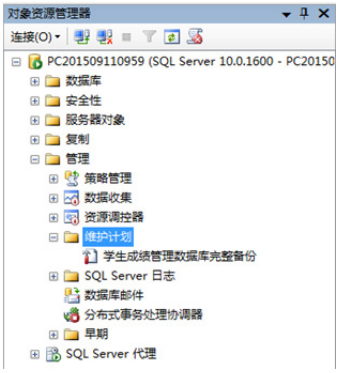


图 3-103　查看建立的维护计划

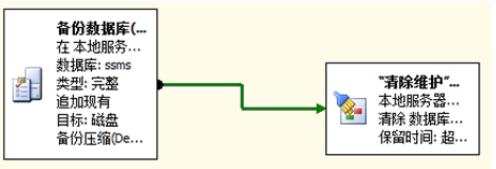


图 3-104　修改维护计划内容

如图 3-105，打开刚才已建立的数据备份计划，从维护计划任务列表中将清除维护任务拖入备份任务所在框；并拖动关联箭头建立备份任务与清除任务两个任务的关联关系。该关联关系用于指定执行的先后顺序。

图 3-105　查看维护计划内容

⑧ 最后重启 SQL Server 代理服务即可。

如上方式是通过“维护计划向导”完成设置，也可以通过“新建维护计划”功能完成，通过新建维护计划，然后拖动备份数据库任务、清楚维护任务，相关设置同上。

（2） 差异备份。

① 选择“对象资源管理器”下的“管理”，对“维护计划”单击右键，在弹出的快捷菜单中双击“新建维护计划”，如图 3-106 所示。

② 在弹出的“新建维护计划”对话框中设置维护计划名称为“学生成绩管理数据库差异备份”，然后，单击“确定”，如图 3-107 所示。

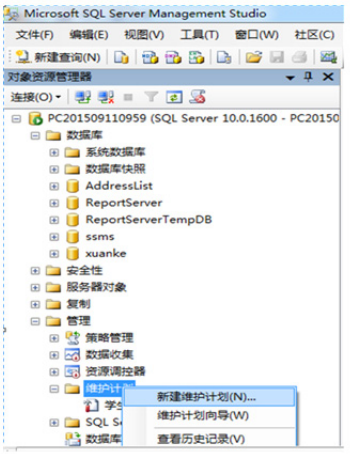


图 3-106　新建维护计划

图 3-107　输入新建维护计划名称

③ 打开差异备份计划，从左侧“工具箱”中拖动“备份数据库任务”和“清除维护任务”到当前计划任务框中，如图 3-108 所示。将两个任务的执行次序定为先执行“备份数据库任务”，再执行“清除维护任务”。

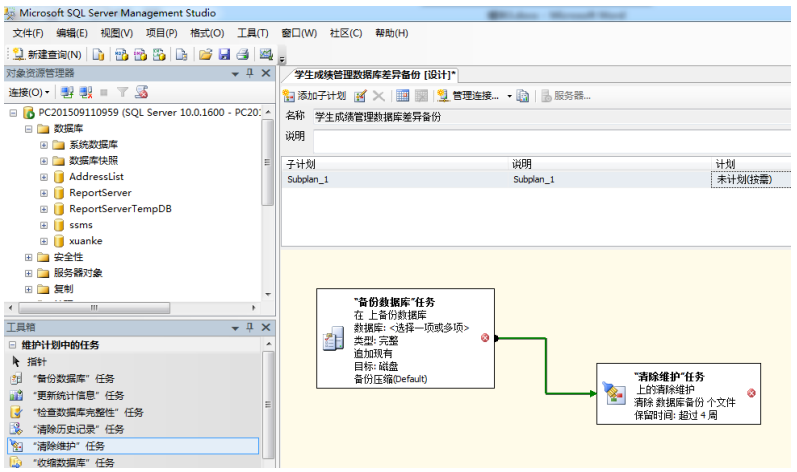


图 3-108　差异备份计划

④ 双击打开“备份数据库”任务，在弹出的对话框中设置备份数据库任务中备份类型为差异，指定需要差异备份的数据库为 ssms 后单击“确定”按钮，返回该对话框后，指定备份存储文件夹为“E：\book”，如图 3-109 所示，最后单击“确定”即可。

⑤ 双击打开“清除维护任务”对话框，设置清除维护任务中文件夹以及文件保留周期如图 3-110 所示。

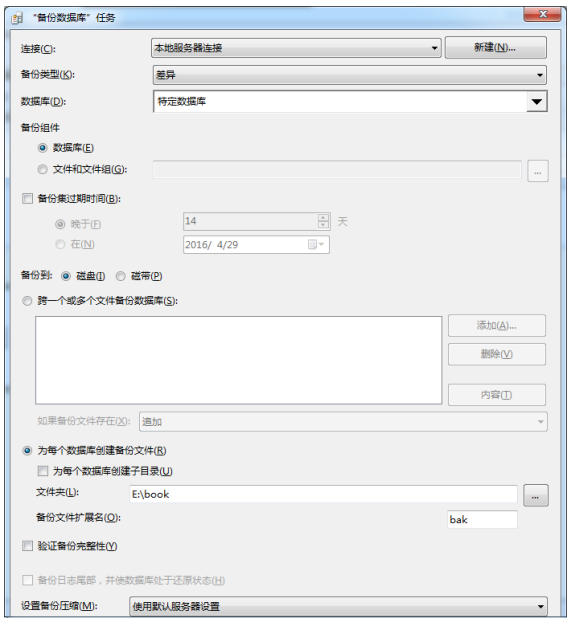


图 3-109　备份数据库任务窗口

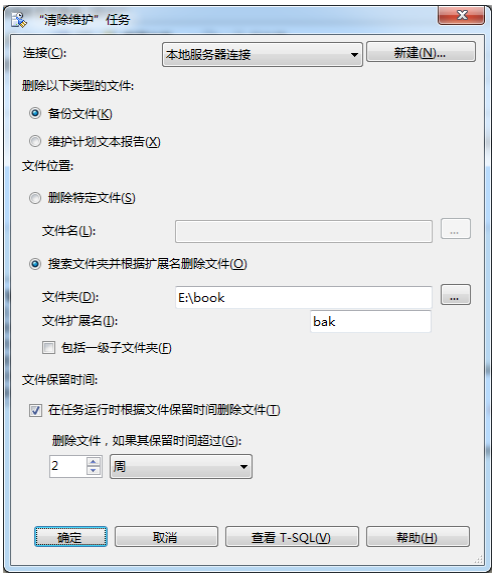


图 3-110　清楚维护任务窗口

注意：两个任务设置完成后，在当前计划任务框中每个任务后面的红色小叉就会消失，说明任务创建成功。创建成功后如图 3-111 所示。

（3） 事务日志备份。

① 选择“对象资源管理器”下的“管理”，对“维护计划”单击右键，在弹出的快捷菜单中双击“新建维护计划”，如图 3-112 所示。

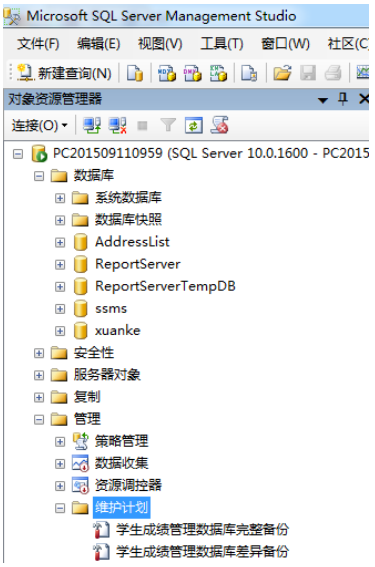


图 3-111　查看创建的差异备份计划

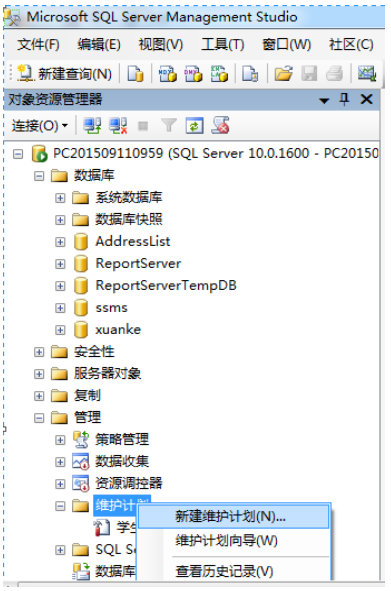


图 3-112　新建维护计划

② 在弹出的“新建维护计划”对话框中设置维护计划名称为“学生成绩管理数据库事务日志备份”，然后，单击“确定”。如图 3-113 所示。

③ 事务日志备份与差异备份类似，其不同之处在于备份类型为“事务日志”。创建成功后，在对象资源管理器的管理中，刷新查看如图 3-114 所示。

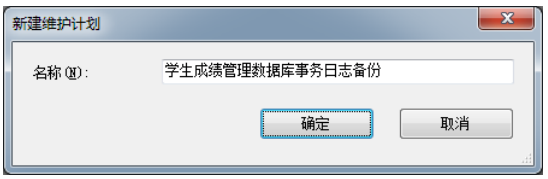


图 3-113　输入维护计划名称窗口

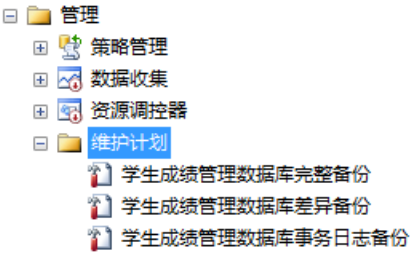


图 3-114　查看创建的事务日志备份计划

2） 手动备份。

手动备份有三种备份方案，即完整备份、差异备份和事务日志备份。由于三种备份过程基本类似，区别仅在于备份类型，因此，此处仅以手动完整备份为例，其他由读者自行完成。

① 在开始菜单中单击“SQL Server Management Studio”，打开 SQL Server 2008“对象资源管理器”，在“对象资源管理器”中选择学生成绩管理数据库 ssms，单击鼠标右键，在弹出的快捷菜单中选择“任务”，继续弹出下一级菜单，从中选择“备份”功能，即对 ssms 数据库进行备份，如图 3-115 所示。

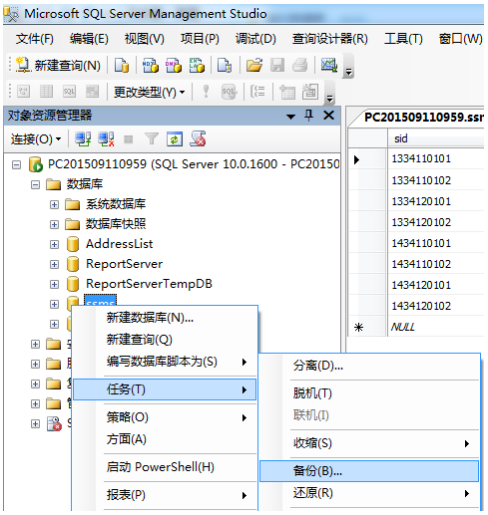


图 3-115　手动备份数据库

单击“备份”功能后，弹出备份数据库对话框，如图 3-116 所示。

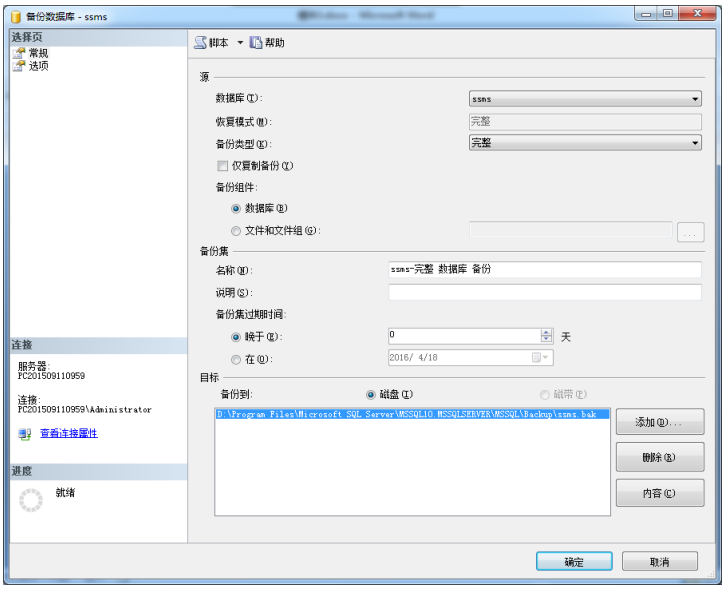


图 3-116　备份数据库对话框

② 其默认的备份数据库为“ssms”，备份类型默认为“完整”备份，注意，在备份数据库对话框的目标中，添加备份位置为“E:\book”目录，点击“添加”按钮出现如图 3-117 所示对话框。

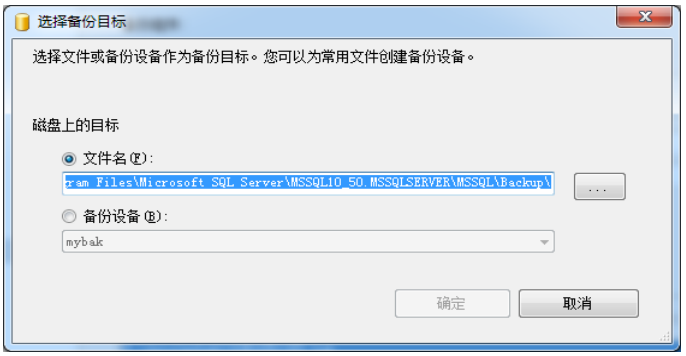


图 3-117　选择备份目标窗口

③ 在选择备份目标中单击文件名后面的“...”选择按钮，弹出定位数据库文件如图 3-118 所示。

④ 在“定位数据库文件”选择备份文件存放路径为“E:\book ”，文件类型采用默认“bak”文件，备份文件名为“ssms20160418 完整.bak”，如图 3-119 所示。注意：一定要带扩展名，否则，最终的备份文件类型不确定，即系统不会加扩展名。

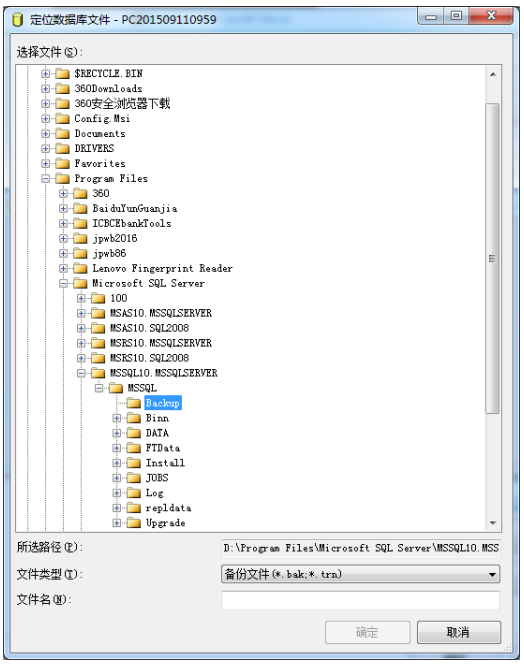


图 3-118　选择备份文件

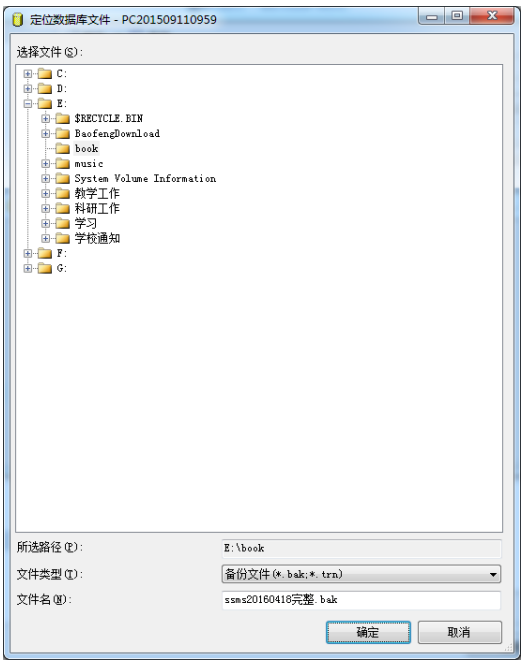


图 3-119　输入备份文件名

⑤ 在图 3-119 中，点击“确定”按钮，将返回到选择备份目标对话框，继续单击“确定”按钮，返回到“备份数据库”对话框，这时在备份目标文件的位置中出现两个位置，删除系统默认位置后单击“确定”，此时，系统开始进行备份，当备份完成时，将弹出如图 3-120 所示的提示信息，同时，查看指定的文件，可以看到备份文件已经存在，如图 3-121 所示。

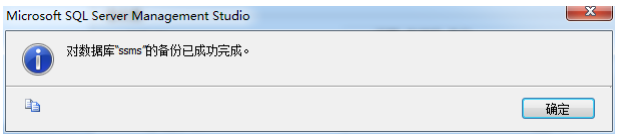


图 3-120　备份完成消息框

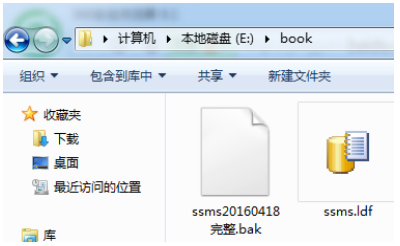


图 3-121　查看备份文件

注意：事务日志备份时需要在在选项中修改内容如图 3-122 所示，其他操作类似。

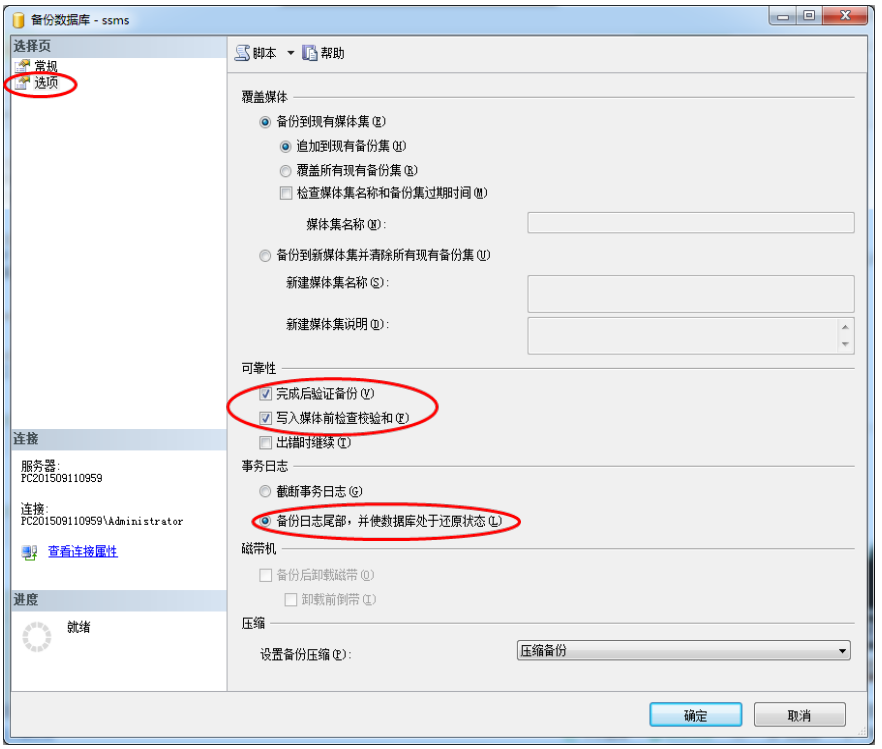


图 3-122　事务日志备份选项

2.　数据库恢复技术

数据库恢复技术主要采用的是利用备份文件对原数据库进行还原，将数据库还原到某一备份时的状态。

（1）　利用完整备份进行还原。

在开始菜单中单击“SQL Server Management Studio”，打开 SQL Server 2008“对象资源管理器”，在“对象资源管理器”中单击“数据库”，弹出相应的快捷菜单，如图 3-123 所示。

在图 3-123 中，选择“还原数据库”功能，将弹出还原数据库对话框，如图 3-124 所

示。在还原数据库对话框的常规选项卡下(默认为常规),设置还原目标数据库名称为“ssms”,还原的源中选择“源设备”,同时,单击其右侧的“…”按钮,弹出“指定备份”对话框,在指定备份对话框中,单击“添加”按钮,弹出“定位备份文件”对话框,在该对话框中,选择用于还原的备份文件,如图 3-125 所示。

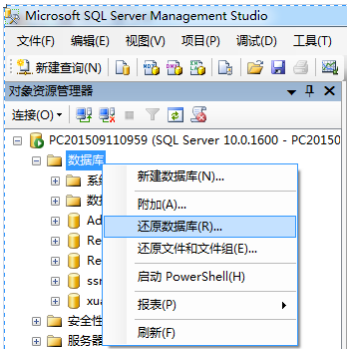


图 3-123　还原数据库

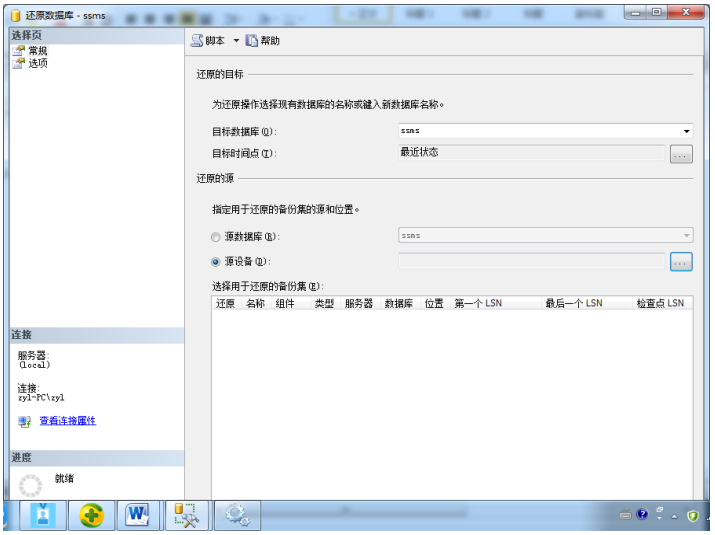


图 3-124　还原数据库对话框

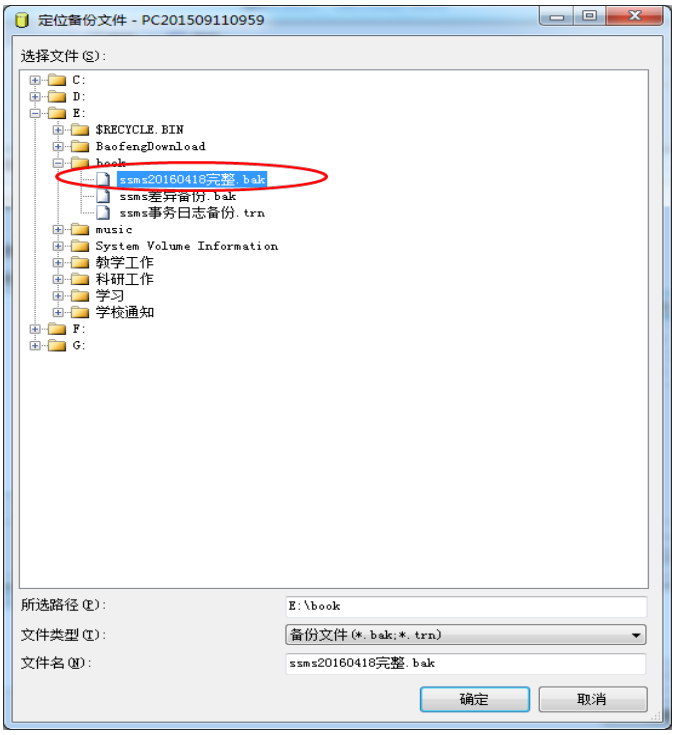


图 3-125　定位要还原的文件

确定还原文件后，单击“确定”按钮，返回指定备份对话框，如图 3-126 所示。

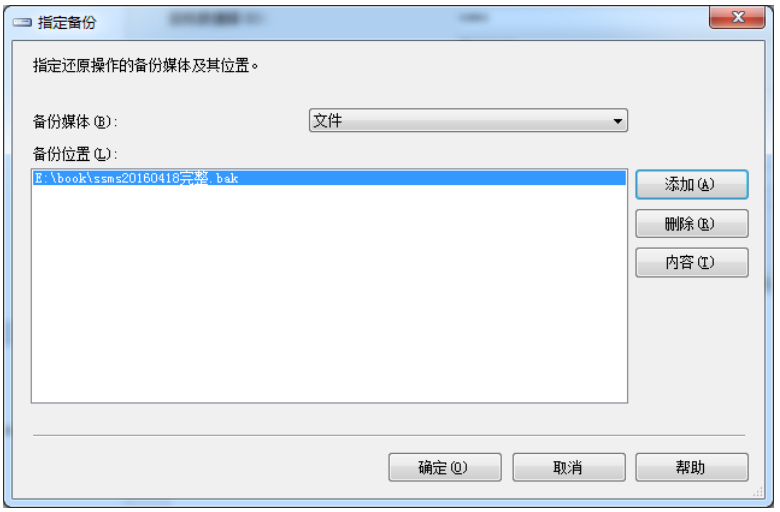


图 3-126　指定备份对话框

在指定备份对话框中，继续单击“确定”按钮，返回还原数据库对话框，如图 3-127 所示。

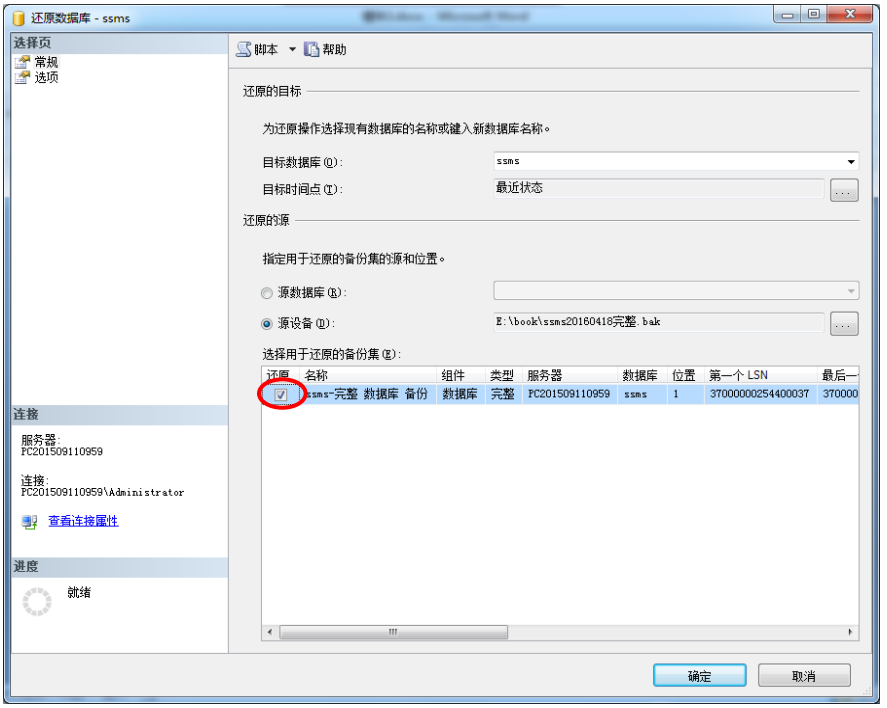


图 3-127　还原数据库对话框

在该对话框中，选择用于还原的备份集中，将备份集前面的复选框勾选上。然后，切换该对话框的标签项为“选项”，如图 3-128 所示。

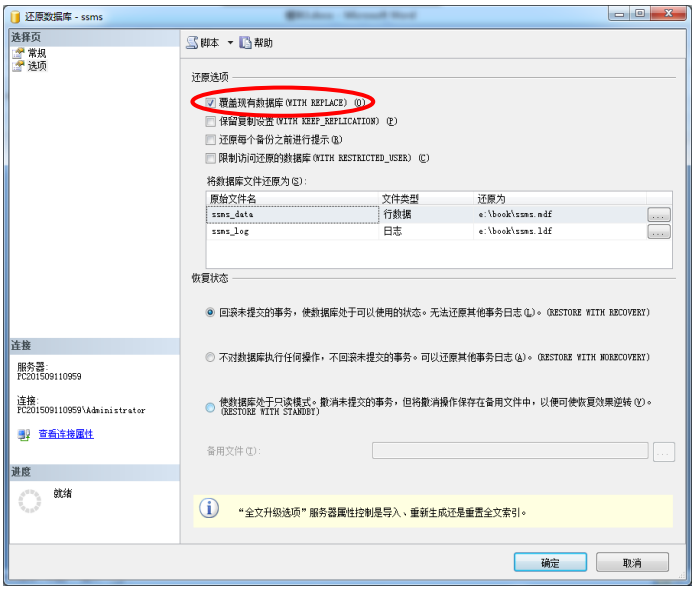


图 3-128　还原数据库选项卡

在该对话框中，设置还原数据库文件存放路径，并选择覆盖现有数据库、回滚未提交事务，最后单击“确定”按钮，将弹出消息框如图 3-129 所示。

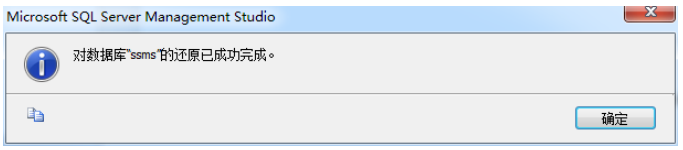


图 3-129　成功还原数据库消息窗口

（2） 利用完整备份+差异备份进行还原。

该方式是将数据库还原到差异备份的状态。在还原完整备份后，可以继续对目标数据库还原差异备份，用于将差异备份保存的数据更新进入当前数据库，使数据库还原到差异备份时的状态。在“对象资源管理器”下，同样右键单击数据库，弹出的快捷菜单如图 3-130 所示。

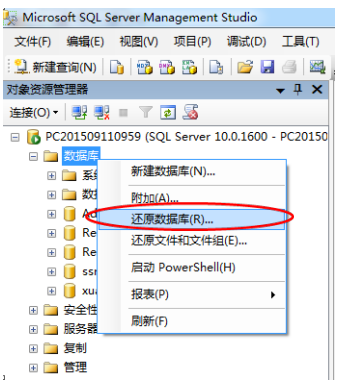


图 3-130　还原数据库快捷菜单

在图 3-130 中，选择“还原数据库”功能，弹出“还原数据库”对话框如图 3-131 所示。

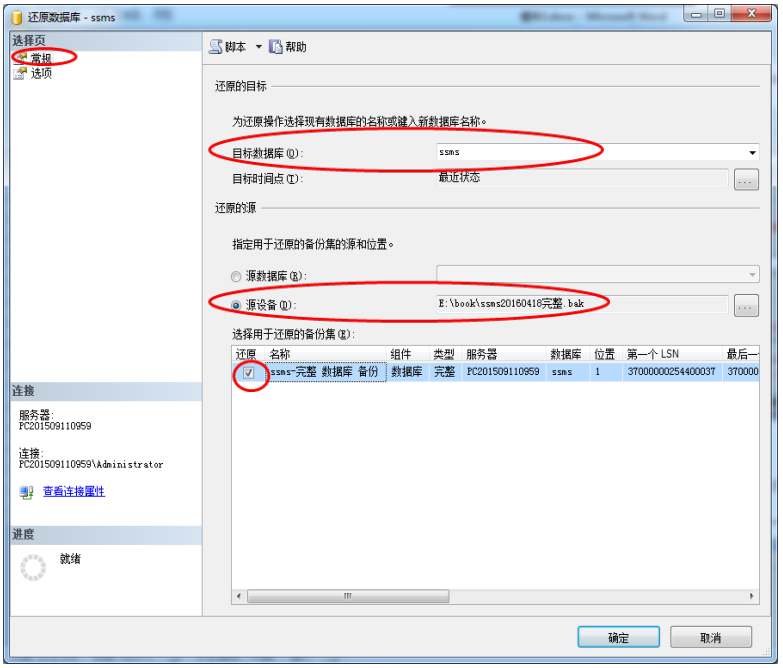


图 3-131　还原数据库对话框

在还原数据库对话框的“常规”选项下，设置还原目标数据库名称，源设备（数据库完备文件）等，然后切换到“选项”对话框下，如图 3-132 所示。

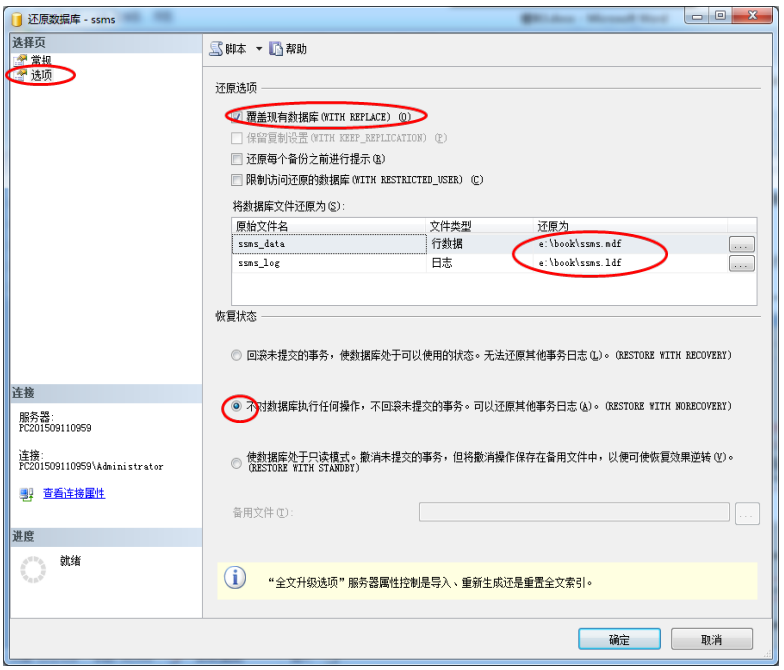


图 3-132　还原数据选项卡

在图 3-132 中，设置还原数据库文件存放路径，并选择覆盖现有数据库、不对数据库执行回滚未提交事务。该步骤完成后，数据库处于还原中状态，继续对数据库进行还原，源设备选择差异备份文件，如图 3-133 所示。

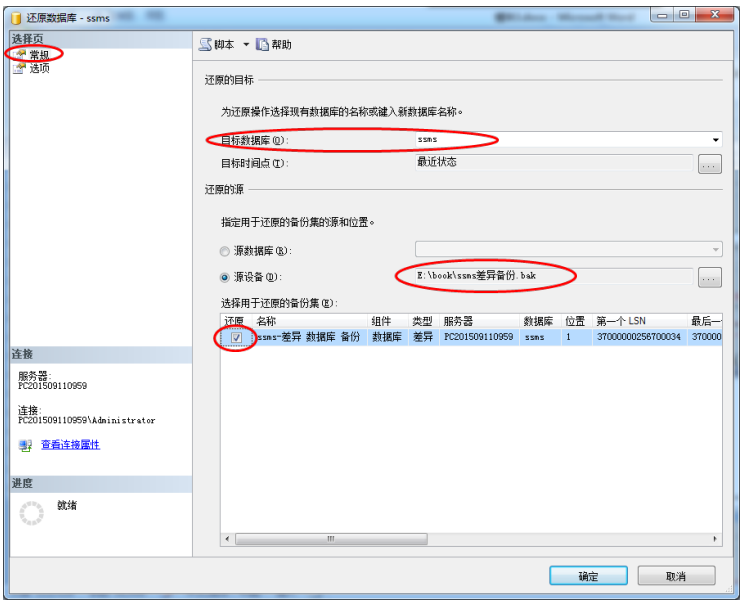


图 3-133　差异备份还原窗口

然后，在还原数据库对话框的“选项”中，设置差异备份文件还原时文件存放路径，数据库事务日志回滚方式，如图 3-134 所示。

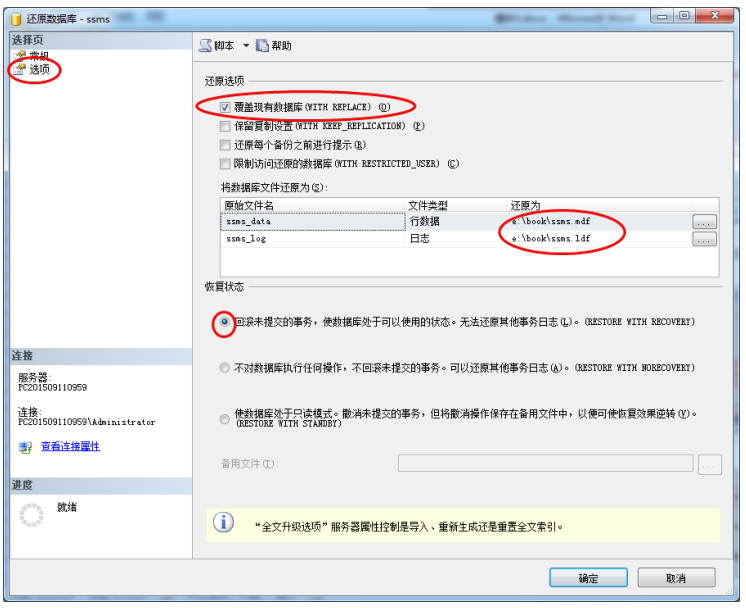


图 3-134　设置差异备份还原选项

（3） 利用完整备份+差异备份+事务日志备份进行还原。

该方式是将数据库还原到事务日志备份时的状态。在还原完整备份后，可以继续对目标数据库还原差异备份，然后再继续还原事务日志备份，用于将差异备份、事务日志备份保存的数据更新进入当前数据库，使数据库还原到事务日志备份时的状态。

在利用完整备份+差异备份还原完成后，数据库处于还原中状态，如图 3-135 所示。

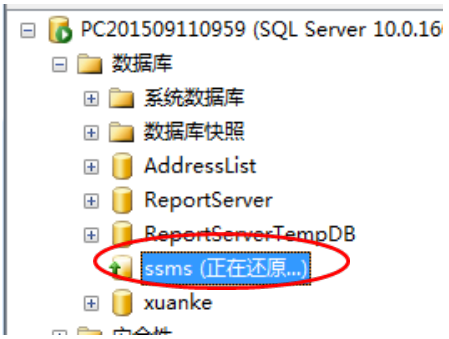


图 3-135　数据库处于还原中状态

在图 3-135 下，继续选择还原，还原事务日志文件到当前数据库。

在图 3-135 中，选择“事务日志”还原选项，如图 3-136 所示，然后会弹出如图 3-137 所示的“还原事务日志”对话框。

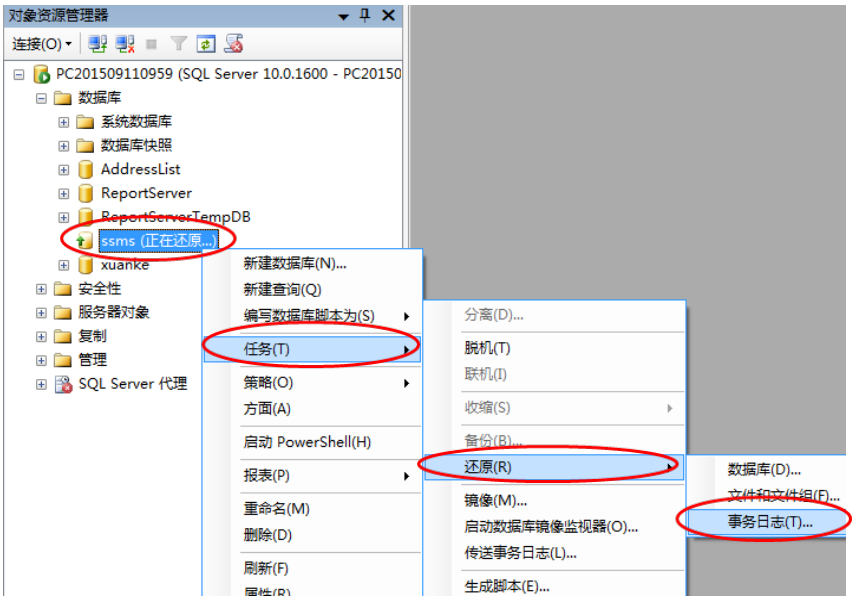


图 3-136　还原事务日志

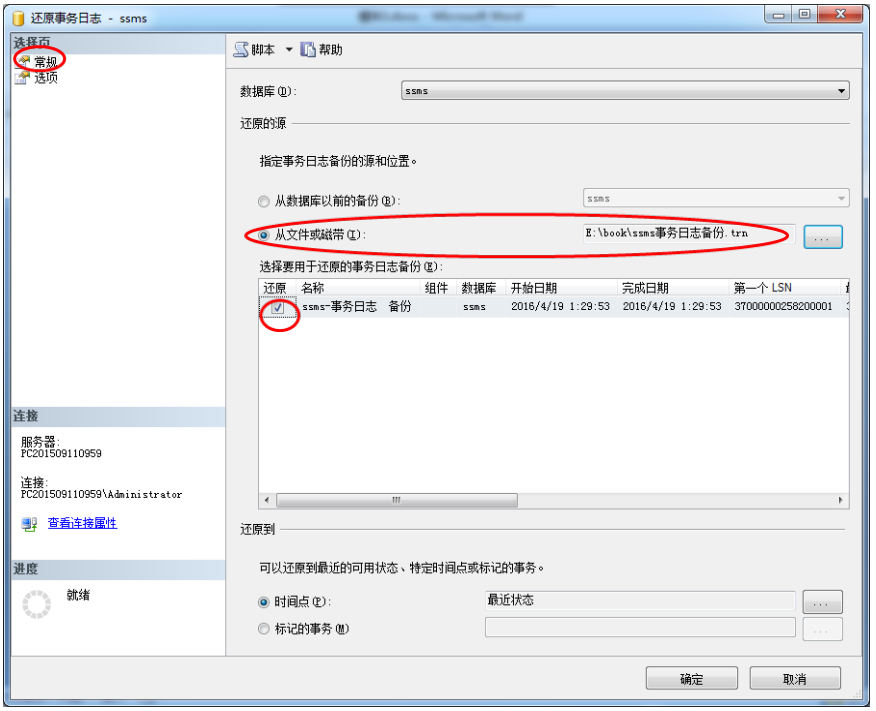


图 3-137　还原事务日志对话框

在图 3-137 中，设置还原的数据库、还原时所用的事务日志文件等信息，同时将指定还原到相应的时间点，如图 3-138 所示。

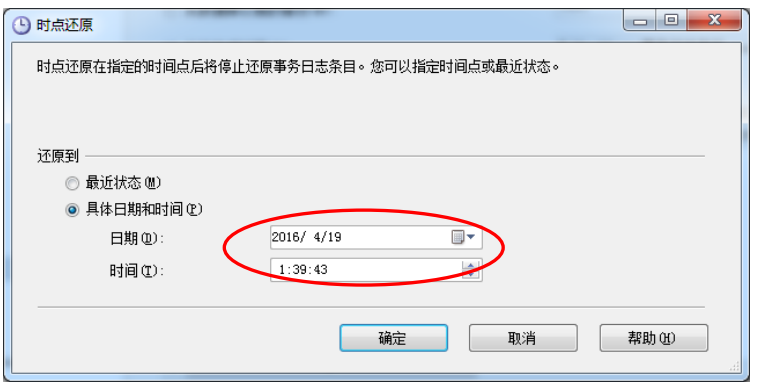


图 3-138　设置还原时间

在图 3-138 设置还原时间对话框中进行还原时间点设置，通过设置该时间点，可以将数据库还原到做完备时至事务日志备份时间段内的任一时间点。然后在“选项”中设置恢复状态为“回滚未提交事务”，如图 3-139 所示。

执行完成后，数据库恢复可用状态，数据更新到指定时间的状态。如果没指定时间，则为事务日志备份时状态。

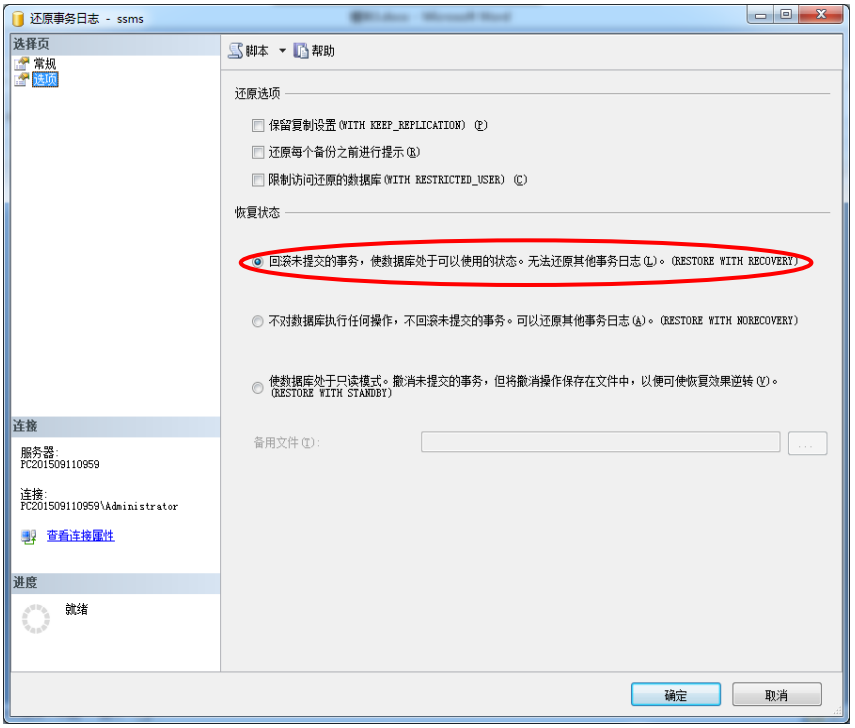


图 3-139　还原事务日志选项设置

习题

根据本案例中的相关知识，制定一个数据库的自动备份方案。

模块 4　学生成绩管理系统优化

学习目标

通过本模块的学习，掌握运用 MVC 设计模式进行系统设计的方法；熟悉采用 MD5 加密算法对用户密码进行加密；掌握使用 JSP 技术通过存储过程和游标访问 SQL Server 2008 数据库的方法；了解 SQL Server 2008 数据库安全控制机制。

主要内容

（1） MVC 设计模式。
（2） MD5 加密算法。
（3） 存储过程的定义和使用。
（4） 游标的定义和使用。
（5） JSP 调用不同形式存储过程的方法。
（6） SQL Server 安全控制机制。

重点与难点

（1） MVC 设计模式。
（2） 存储过程的定义。
（3） SQL Server 安全控制机制。

案例 1　开发MVC模式的学生成绩管理系统

通过前面的学习已经完成了学生成绩管理系统的设计与实现，但是这种设计方式，存在很大的弊端，整个系统开发没有分层的概念，其业务逻辑代码、事务控制代码和视图代码均写在一起，通过前面的系统调试过程不难发现，将 HTML、JSP、JavaScript 和 JDBC 代码编写到一起，这样的程序结构复杂、调试困难，这样的开发方式不利于软件的维护及扩展，更不能实现代码的重用，也不利于团队的协作开发。采用 MVC 设计模式，可以让业务逻辑（模型层 M）、界面设计（视图层 V）和数据处理（控制层 C）各处独立，分别由不同的设计人员来完成，最后由团队协作完成整个应用系统的设计工作。遵循 MVC 设计思想，降低了程序中的耦合，如果业务逻辑及业务计划有所改变，则只需要修改模型层代码即可，而视图层和控制层代码不需要修改，这样能使 Web 程序具有一定的健壮性和可

扩展性。

任务 1　创建不同模式对象的包

MVC 设计模式把 Web 应用程序分为 3 个核心模块：模型（Model）、视图（View）和控制器（Controller）。这三个模块中的对象位于不同的包中，通常包括控制器所在的包 server 包，实体所在的包 entity 包，模型所在的包，模型包又包括数据载体包 dao 包和业务模型包 biz 包。包命名规范是：包名小写，并且结构倒置，即按照如下结构创建 MVC 模式包：biz、bizimpl、dao、daoimpl、entity、server。建立包步骤如下：

（1） 建立 Web 项目，具体操作步骤同模块三中新建 Web 项目。打开 Eclipse，单击“文件”菜单中的“新建”菜单项，然后选择“动态 Web 项目”。输入项目名称 studentScorePro，单击“下一步”，再单击“下一步”，选中“Generate web.xml deployment descriptor”复选框。然后单击“完成”按钮，完成项目的创建。如图 4-1 所示。

（2） 在建立的 Web 项目 studentScorePro 下找到“JavaResources”文件夹，再在其中找到“src”文件夹，在“src”上单击鼠标右键选择“新建”，再在新建菜单下选件“包”如图 4-2 所示。

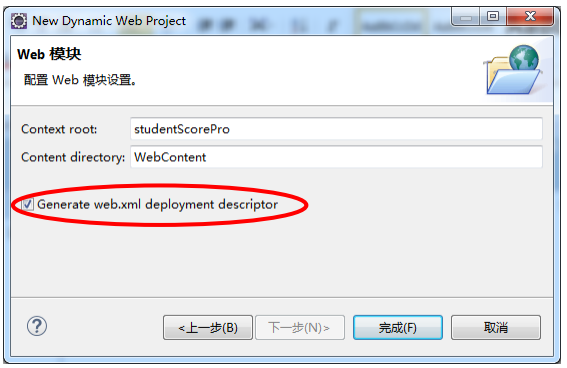


图 4-1　新建动态 Web 项目对话框

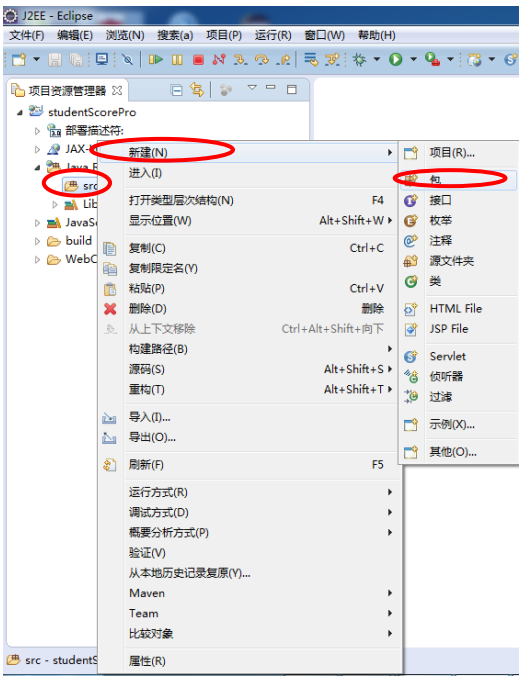


图 4-2　新建包操作窗口

（3） 在弹出的新建 Java 包窗口中输入控制器 servlet 所在的包名 com.ssms.server 如图 4-3 所示。

在“新建包”窗口中单击“完成”按钮，完成了控制器所在的包的创建，如图 4-4 所示。

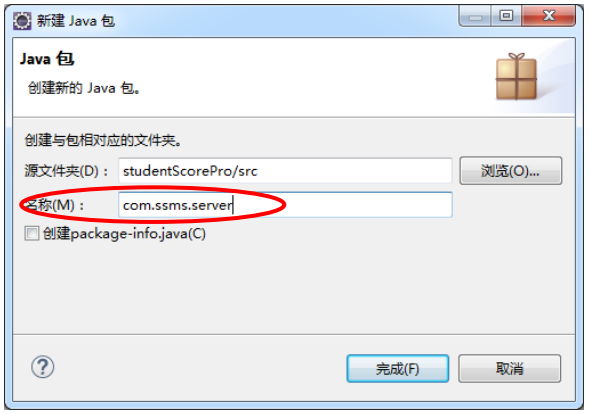


图 4-3　新建包窗口

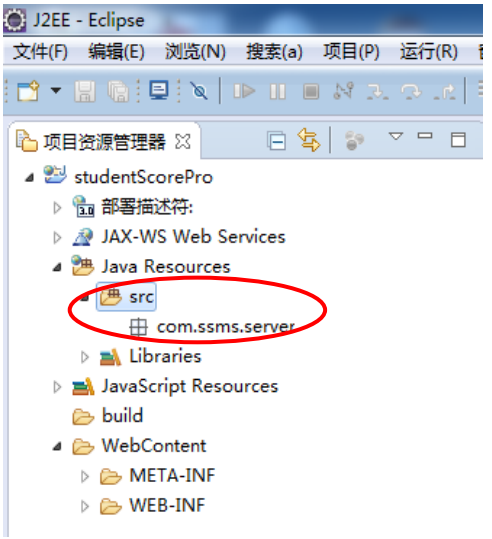


图 4-4　创建的 server 包

（4）　按照同样的步骤分别建立实体所在的包 com.ssms.entity 和模型所在的包 com.ssms.dao、com.ssms.dao.impl 和 com.ssms.biz、com.ssms.biz.impl，结果如图 4-5 所示。

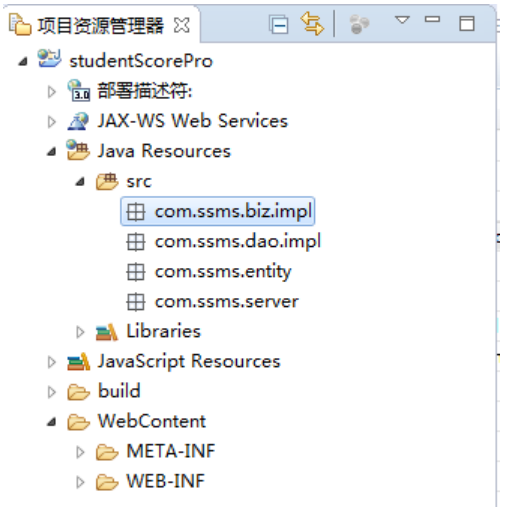


图 4-5　项目建立的所有包

任务 2　创建实体对象

首先创建 student 对象，具体操作步骤如下：

（1）　在上面创建的实体对象包“com.ssms.entity”中单击鼠标右键，在弹出的菜单中选择“新建”，然后再选择“类”菜单项，如图 4-6 所示。

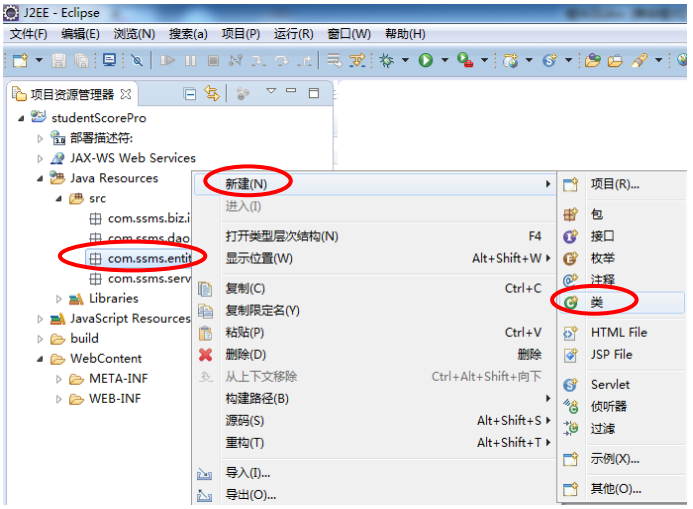


图 4-6　新建类窗口

在弹出的新建 Java 类窗口的“名称框”中输入 Student 类名，然后单击“完成”按钮，如图 4-7 所示。

注意要把 Student 类放在 com.ssms.entity 包中，如果弹出的窗口，显示的包不是 com.ssms.entity，单击“浏览”按钮会弹出如图 4-8 所示的选择包窗口，选择类所在的包名，然后单击“确定”按钮，会进入到 student 类所在的编辑界面。

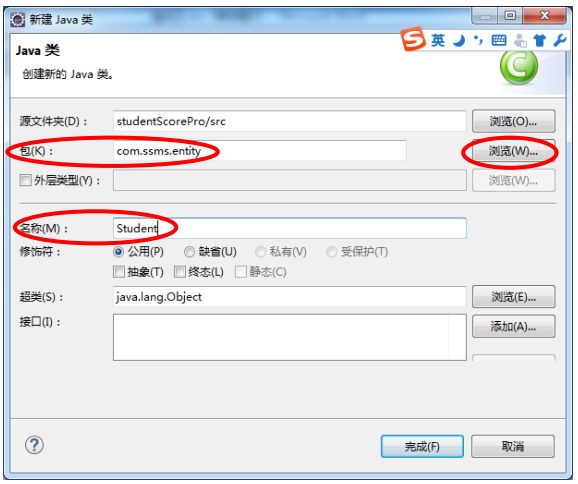


图 4-7　新建 Java 类窗口

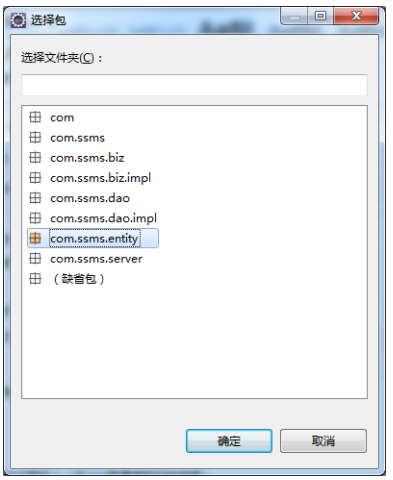


图 4-8　选择类所在的包窗口

（2） 在 Student 类的编辑界面，输入如下代码，给 Student 类添加属性：

```
private String sid;
    private String sPassword;
    private String sName;
    private String sClassName;
    private String sSex;
```

```
    private Date sBirthday;
    private String sPolitical;
    private boolean fresh;
    private String sPhotoPath;
    private String sEmail;
```

（3） 导入 Date 类所在的包。

因为代码中用到“Date”类，这时“Date”类下面会出现红色错误提示，需要导入 Date 类所在的包，有两种方式可以完成包的导入，第一种方式是：光标移动到 Date 位置，选择“源码”菜单的“添加导入”菜单项，如图 4-9 所示。

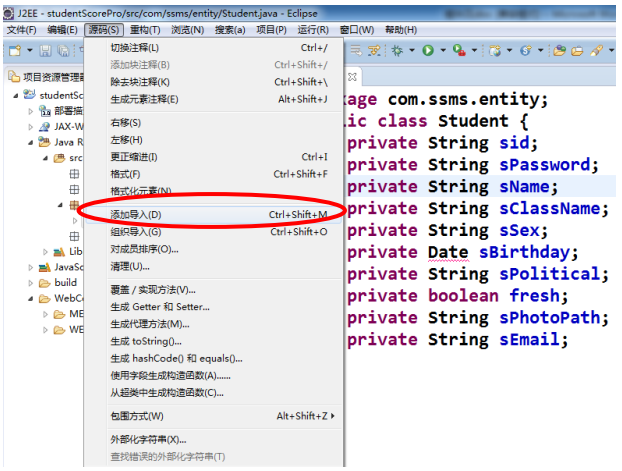


图 4-9　通过源码菜单添加导入

在弹出的添加导入窗口中选择 java.util.Date 包，如图 4-10 所示，然后单击“完成”按钮，完成 Date 包的导入。第二种方式是：把光标移动到 Date 位置，会自动弹出一个快速修正提示框，如图 4-11 所示，选择其中的“导入“Date”（java.util）”选项，完成 Date 包的导入。

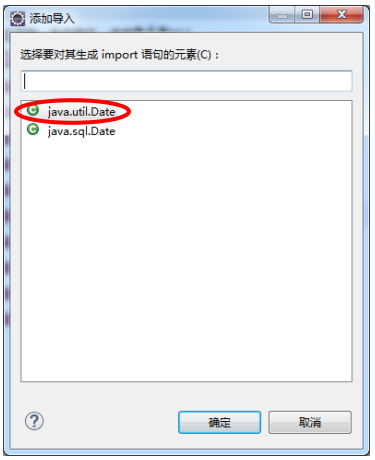


图 4-10　选择添加导入的包

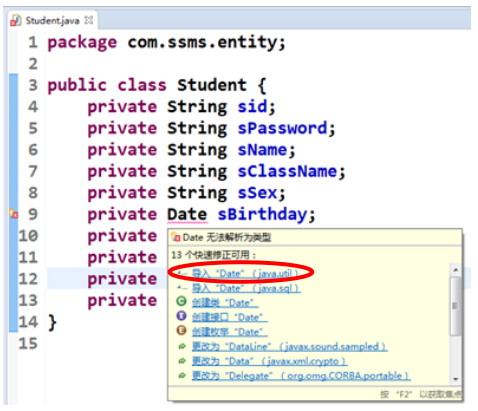


图 4-11　通过快捷菜单导入包

（4） 为 Student 类添加 get 和 set 方法。

在项目资源管理器中找到建立的 Student 类，鼠标右键单击该类名，会弹出一个快捷菜单，如图 4-12 所示。

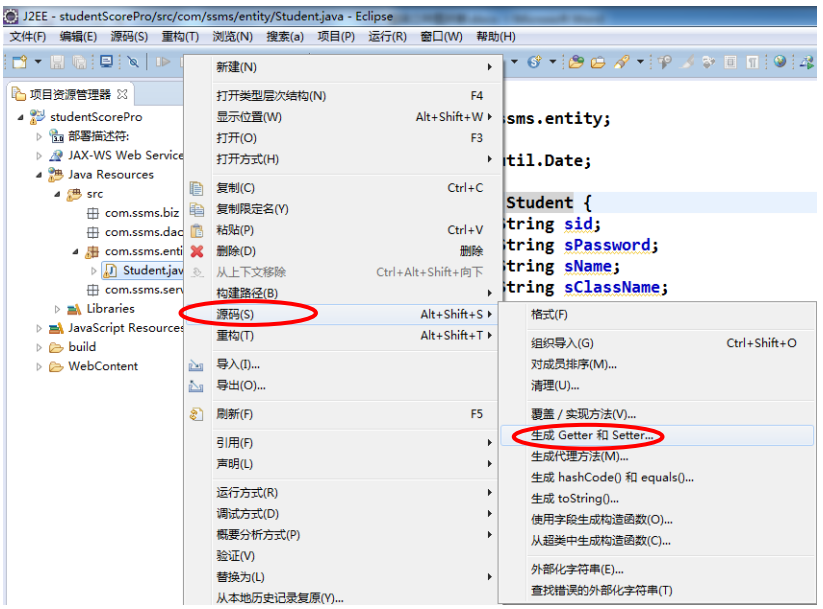


图 4-12　生成 Getter 和 Setter 方法

在图 4-12 中选择快捷菜单中的“源码”，然后再选择其中的“生成 Getter 和 Setter……”菜单项，然后会弹出生成 Getter 和 Setter 的窗口，如图 4-13 所示。

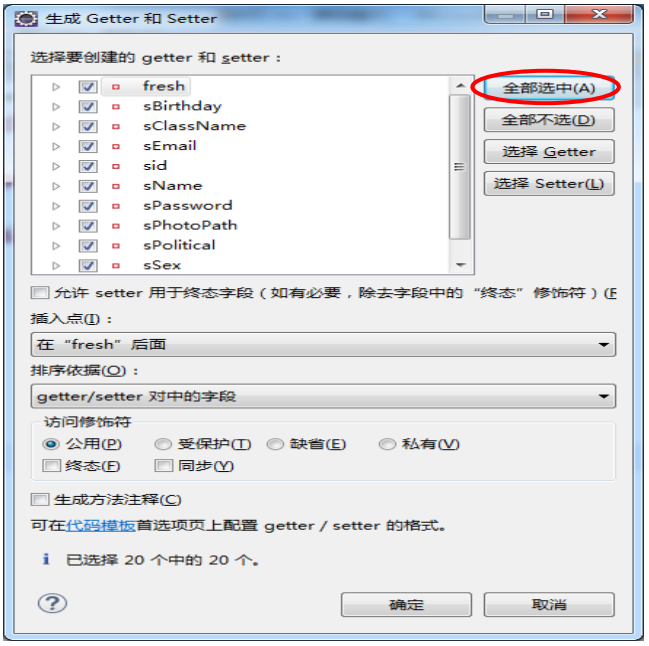


图 4-13　选择生成 Getter 和 Setter 方法的属性

单击“全部选中”按钮，选择对所有属性建立 Getter 和 Setter 方法，然后单击“确定”完成对 Student 类的创建。依据模块三中设计的数据库表中的字段，按照同样的方法和步骤完成其他对象的创建。结果如图 4-14 所示。

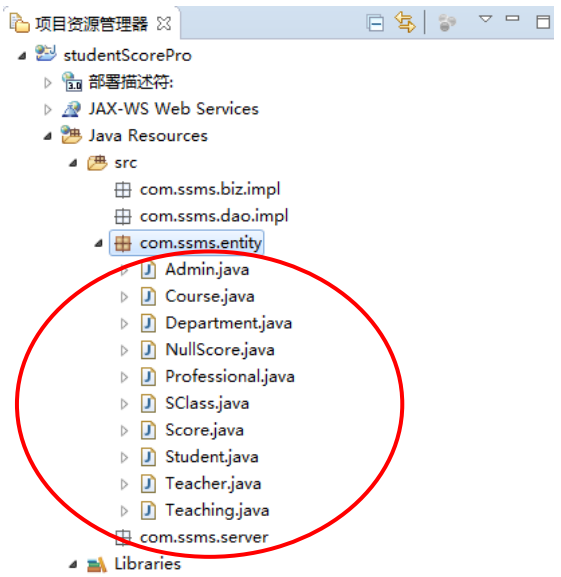


图 4-14　项目建立的实体类

任务 3　创建数据处理模型

建立数据处理模型，实现数据库的连接和数据操作对象的关闭操作，具体操作步骤如下：

按照前面介绍的创建 Student 类的方法和步骤，在 com.ssms.dao 包中新建 BaseDao 类，具体代码如下：

```
   package com.ssms.dao;
   import java.sql.Connection;
   import java.sql.DriverManager;
   import java.sql.PreparedStatement;
   import java.sql.ResultSet;
   import java.sql.SQLException;
   /*
    * 静态类
    * 此类负责数据库的连接与关闭
    * */
   public class BaseDao {
      public static Connection getConn() {
            Connection conn = null;
   try {Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");//加载
SQL Server 驱动
                  conn = DriverManager.getConnection(
     "jdbc:sqlserver://localhost:1433;DatabaseName=ssms", "sa", "hh123456");}
```

```
catch (ClassNotFoundException e) {
                    e.printStackTrace();
             }
                     catch (SQLException e) {
                    e.printStackTrace();
             }
             return conn;
      }
      public  static  void  closeAll(ResultSet  rs,PreparedStatement  pstmt,
Connection conn){
      try {if(rs !=null)
             {rs.close();}
      } catch (SQLException e2) {
             e2.printStackTrace();      }
      try {if(pstmt!=null)
             {pstmt.close();}
      } catch (SQLException e1) {
             e1.printStackTrace();      }
      try {if(conn!=null)
             {conn.close();}
      } catch (SQLException e) {
             e.printStackTrace();
      }
      }
   }
```

任务 4　实现学生成绩管理系统的用户登录

一、建立登录页面

按照模块三中建立 JSP 页面的方法和步骤，在 WebContent 目录下建立 login.jsp 页面，其代码如下：

```
   <%@ page language="java" pageEncoding="UTF-8"%>
   <title>登录页面</title>
   <head>
   </head>
   <script type="text/javascript">
      function checksubmit() {
             with (document.logform) {
                    var uid = userid.value;
                    var pwd = upass.value;
                    if (uid == null || uid == "") {
                           alert("请填写登录账号 !");
                           return;//函数返回
                    } else if (pwd == null || pwd == "") {
                           alert("请填写密码!");
                           return;
                    } else
```

```
                        document.logform.submit();
            }
        }
    </script>
    <body>
        <H2 align="center">请登录</H2>
        <hr>
        <form action="Login" method="post" name="logform">
            <table border="1" class="font" align="center">
                <tr>
                    <td align="center">身  份: </td>
                    <td><select name="logintype" style="width: 160px">
                            <option value="teacher">老师</option>
                            <option value="admin">管理员</option>
                    </select></td>
                </tr>
                <tr>
                    <td align="center">用户账号: </td>
            <td><input type="text" name="userid" style="width: 160px" /></td>
                </tr>
                <tr>
                    <td align="center">密  码: </td>
        <td><input type="password" name="userpass" style="width: 160px" /></td>
                </tr>
                <tr>
                    <td align="center"><input type="submit" value="登
录"
                        style='font-size:      14px'      onClick=
"checksubmit()" /></td>
                    <td  align="center"><input  type="reset"  style=
'font-size: 14px'
                        value="清空"></td>
                </tr>
            </table>
        </form>
    </body>
    </html>
```

login.jsp 中的代码和模块三案例 5 学生成绩管理系统的备份与恢复中 login.jsp 基本相同，只有一点更改，就是 form 表单的 action 属性的值改成了 Login，Login 是一个 servlet 控制器，完成 JSP 和业务处理模型之间进行数据传递及页面跳转控制处理功能。

二、建立教师业务模型

1.　建立教师数据层接口 TeacherDao

在 com.ssms.dao 包中建立 TeacherDao 接口，指定教师要实现的功能。具体操作步骤如下：在 com.ssms.dao 包上单击鼠标右键选择“新建”，然后再选择其中的“接口”如图 4-15 所示。

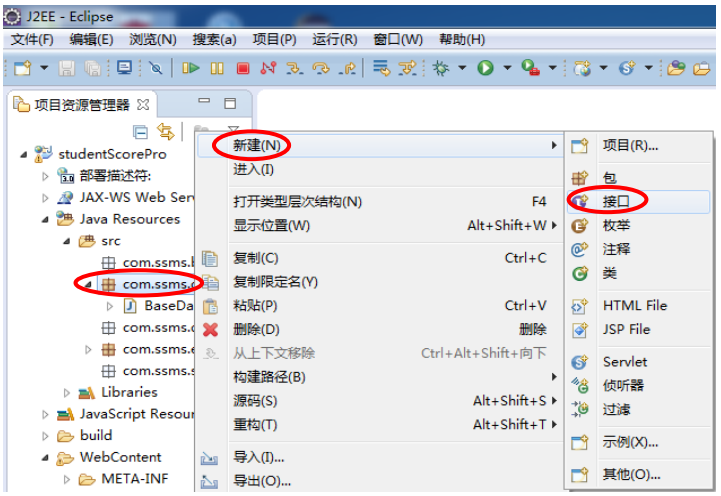


图 4-15　建立接口

选择新建接口后会弹出如图 4-16 所示的新建接口对话框。

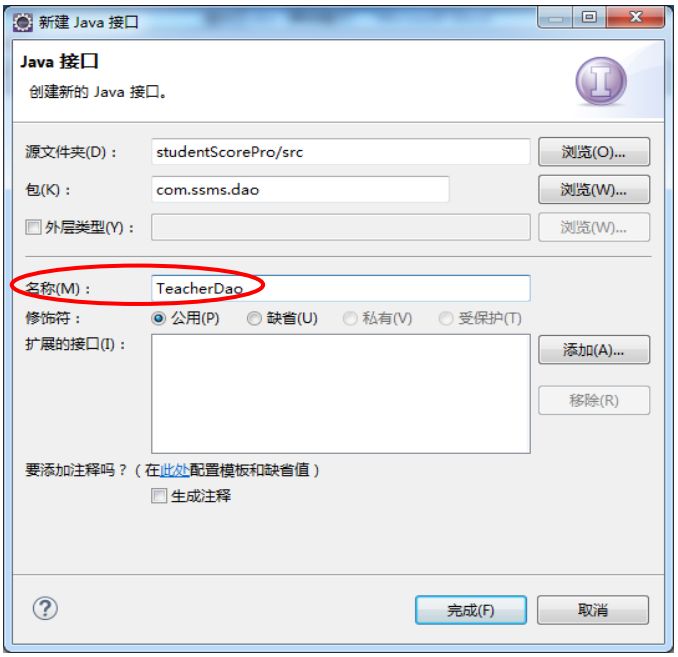


图 4-16　新建 Java 接口

在新建 Java 接口对话框的名称框中输入“TeacherDao”，然后单击“完成”按钮，进入 TeacherDao 接口的编辑界面。在“public interface TeacherDao {”和“}”之间加入如下代码：

```
    public boolean isLogin(String tid,String tpws);//抽象方法用于判定教师是否登录
    public Teacher getOneTeacherById(String id); //抽象方法根据教师得到一个教师对象
```

然后按照上面导入包的方法，导入 Teacher 类所在的包。

2.　建立教师数据层实现类 TeacherDaoImpl

（1）　创建 TeacherDaoImpl 类。

在 com.ssms.dao.impl 包上单击鼠标右键选择“新建”，然后再选择其中的“类”，如图 4-17 所示。

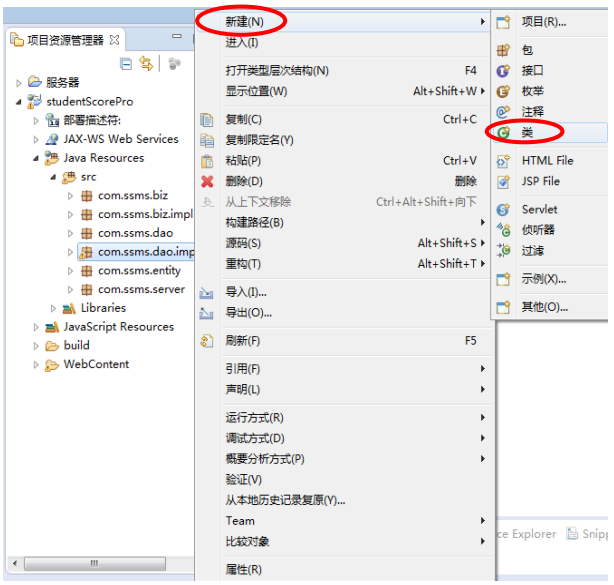


图 4-17　新建类

选择了新建类后会弹出如图 4-18 所示的新建类窗口。

图 4-18　新建 Java 类窗口

在“新建 Java 类”对话框的名称框中输入“TeacherDaoImpl”，然后单击窗口右侧的添加按钮，添加该类实现的接口，弹出如图 4-19 所示的窗口。

在图 4-19 中“匹配项”列出项目中已经建立的接口，如果没有显示相关接口则在“选择接口”中输入接口的相关信息，则已经建立的接口会显示出来，选择要实现的接口，如图 4-19 所示，然后单击“确定”按钮。则又回到图 4-18 所示窗口，这时接口对应框中会添加选择的接口。如图 4-20 所示。单击“完成”按钮，会进入 TeacherDaoImpl 类的编辑界面。

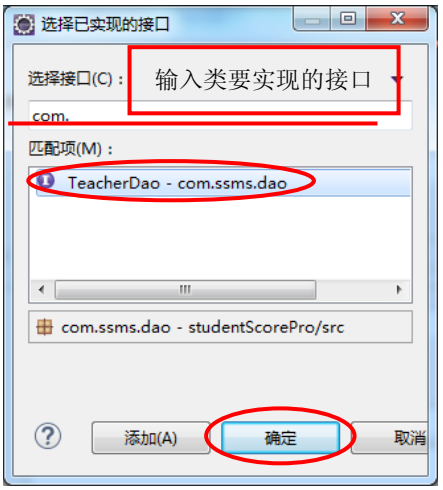


图 4-19　选择类实现的接口

图 4-20　选择了接口后的新建类窗口

（2） 编写 IsLogin 方法。

在 TeacherDaoImpl 类的编辑界面中，编写 IsLogin 方法。在前面登录验证页面 checklogin.jsp 中，虽然已实现了用户登录身份的验证，但是程序中存在大量拼接产生的 SQL 语句的代码，这样存在很大安全隐患，容易遭受 SQL 注入攻击。为了提高程序的安全性，避免 SQL 注入，需要使用参数化的 SQL 语句。即把原来的 SQL 指令

```
   sql = "select * from teacher where tid= '" + uid + "' and  tPassword= '" +
upass + "'"
```

改为：

```
   sql="select * from  teacher where tid=? and tpassword=?";
```

IsLogin 方法中具体代码如下：

```
      Connection conn=null;
      PreparedStatement pstmt=null;
      ResultSet rst=null;
      boolean ret=false;
```

```
    conn=BaseDao.getConn();
    String sql="select * from  teacher where tid=? and tpassword=?";
    try { pstmt=conn.prepareStatement(sql);
          pstmt.setString(1,tid);
          pstmt.setString(2, tpws);
          rst=pstmt.executeQuery( );
           if(rst.next()){ret=true;}
    } catch (SQLException e) {
          e.printStackTrace();
    } finally {
          BaseDao.closeAll(rst, pstmt, conn);
    }
    return ret;
```

因为代码中用到的一些类没有导入相关的包，所以在加入以上代码后会出现一些红色下划线提示要导入相关的包，按照模块四任务二介绍的导入包的方法，导入相关的包选项，完成所需包的导入，红色错误提示即可消失，完成了 IsLogin 方法的编写。

（3） 编写 getOneTeacherById 方法。

在 getOneTeacherById 方法体中加入如下代码：

```
Teacher t=new Teacher();
          Connection conn=null;
          PreparedStatement pstmt=null;
          ResultSet rst=null;
       conn=BaseDao.getConn();
       String sql="select * from teacher where tid=?" ;
    try {pstmt=conn.prepareStatement(sql);
          pstmt.setString(1, id);
          rst=pstmt.executeQuery();
          if(rst.next()){
                t.setTid(rst.getString("tid"));
                t.setTName(rst.getString("tname"));
                t.setTPassword(rst.getString("tPassword"));
          }
          } catch (SQLException e) {
          e.printStackTrace();
    } finally {
          BaseDao.closeAll(rst, pstmt, conn);
    }
          return t;
```

3. 建立教师业务逻辑层

（1） 建立教师业务逻辑层接口。

按照前面建立接口的方法，在 com.ssms.biz 包中建立业务逻辑层接口 TeacherBiz。其代码如下：

```
package com.ssms.biz;
import com.ssms.entity.Teacher;
```

```
public interface TeacherBiz {
public boolean isLogin(String tid,String tpws);
public Teacher getOneTeacherById(String id);
}
```

（2） 建立教师业务逻辑层实现类。

按照前面建立类的方法，在 com.ssms.biz.impl 包中建立业务逻辑层实现类 TeacherBizImpl，该类实现 TeacherBiz 接口。其代码如下：

```
package com.ssms.biz.impl;
import com.ssms.biz.TeacherBiz;
import com.ssms.dao.TeacherDao;
import com.ssms.dao.impl.TeacherDaoImpl;
import com.ssms.entity.Teacher;
public class TeacherBizImpl implements TeacherBiz {
    public boolean isLogin(String tid,String tpws) {
           TeacherDao tdao = new TeacherDaoImpl();
           return tdao.isLogin(tid,tpws);
    }
    public Teacher getOneTeacherById(String id) {
           TeacherDao tdao = new TeacherDaoImpl();
           return tdao.getOneTeacherById(id);
    }
}
```

相关知识

SQL 注入攻击简介

SQL 注入攻击是黑客对数据库进行攻击的常用手段之一。如果程序员编写代码的时候，没有对用户输入数据的合法性进行判断，使应用程序存在安全隐患。黑客通过构造巧妙的 SQL 语句，实现无账号登录系统，或根据程序返回的结果，获得某些他想要的数据，这就是所谓的 SQL Injection，即 SQL 注入。一些黑客通过 SQL 注入检索他人的数据，改变服务器的设置，或者有意破坏他人的服务器。SQL 注入式攻击不是 SQL Server 的问题，而是程序设计不当造成的。

例如在前面编写的 checklogin.jsp 页面，用来对登录用户输入的账号和密码进行验证，其中对教师用户登录验证的 SQL 语句是：sql = "select * from teacher where tid= '" + uid + "' and tPassword= '" + upass + "'"，这样的 SQL 指令很容易进行 SQL 注入。运行前面建立的 studentScorePro 项目，在登录窗体的用户名和密码框中都输入“'1'or'1'=1”，如图 4-21 所示，单击“登录”按钮，发现采用其名称与密码一样能成功登录该系统，其实上面输入的用户名和密码数据库中是根本不存在的，为何能绕过登录验证，而成功登录呢？从上面的 SQL 语句中可以看出，由于表达式'1'='1'的值永远为 true，所以即使输入的用户名不正确，表达式“tid='1'or'1'='1'”的值仍然为 true，所以用户能成功登录系统。那么如何来避免 SQL 注入，避免 SQL 注入有多种方法，下面介绍其中一种方法，采用预编译语句集，它内置了处

理 SQL 注入的能力，只要使用它的 setXXX 方法传值即可。其原理是：SQL 注入只对 SQL 语句的准备（编译）过程有破坏作用，而 PreparedStatement 已经准备好了，执行阶段只是把输入串作为数据处理，而不再对 SQL 语句进行解析，准备，因此也就避免了 SQL 注入问题。

图 4-21　SQL 注入方式登录

把前面拼凑 SQL 语句改为参数化的 SQL 语句

```
    sql="select * from  teacher where tid=? and tpassword=?"JDBC 的语句对象
PreparedStatement 通过给参数赋值
    pstmt.setString(1,tid);
    pstmt.setString(2, tpws);
```

来构造 SQL 语句，避免了 SQL 注入。按照上面 IsLogin 方法中的代码进行用户登录验证，再次运行，在用户账号和密码框中再次都输入“1'or'1'='1”，弹出如图 4-22 所示的错误提示窗口。用户通过构造的 SQL 语句已经不能通过登录验证，无法非法登录。

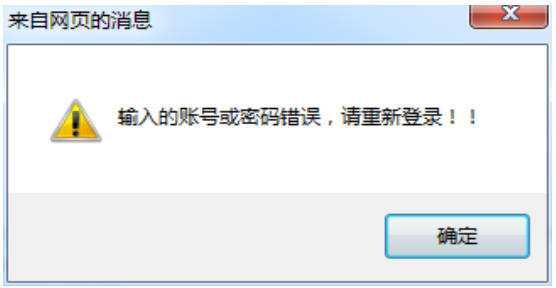


图 4-22　输入账号或密码错误提示窗口

任务 4.1　建立登录控制器

建立一个名为 Login 的 servlet 实现登录信息的读取，并调用数据处理类，完成登录功能，具体操作步骤如下：

在 com.ssms.server 包上单击鼠标右键，在弹出的快捷菜单中选择“新建”，弹出新建的下一级菜单，在其中选择“servlet”，如图 4-23 所示。

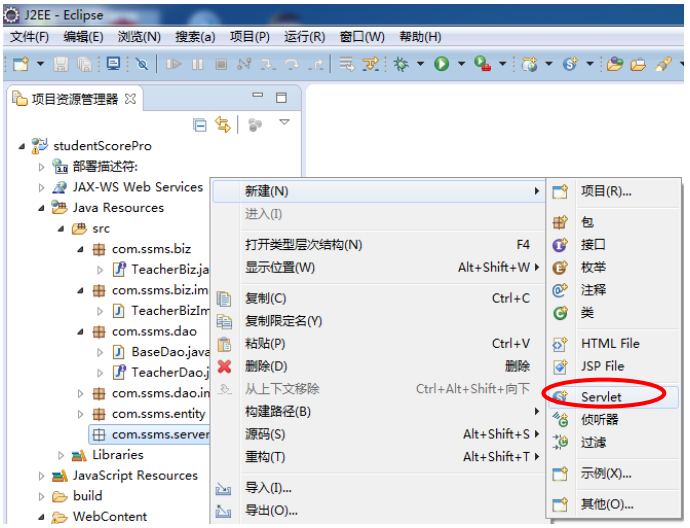


图 4-23　创建 Servlet

单击创建的类型为“servlet”后，弹出创建 servlet 的对话框，如图 4-24 所示。

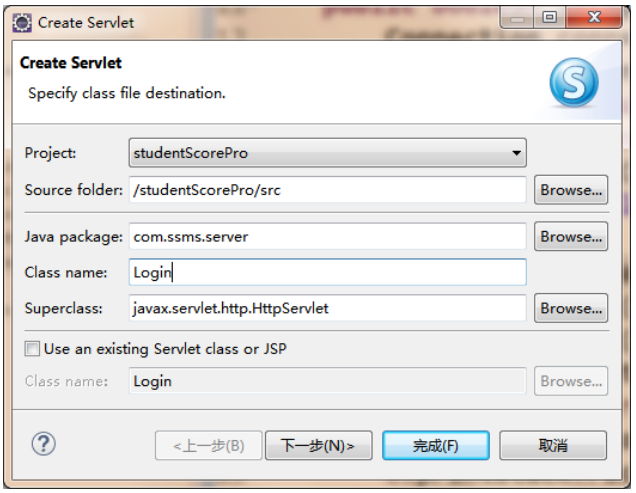


图 4-24　创建 servlet 窗口

在“创建 servlet 窗口”的 class name 对应文本框中输入 servlet 的名字 Login，然后单击“完成”按钮，进入 servlet 的编辑界面。

在 servlet 的 doPost 方法中加入如下代码：

```
request.setCharacterEncoding("UTF-8");
response.setCharacterEncoding("UTF-8");
response.setContentType("text/html;charset=UTF-8");
//接受客户端参数
String uid = request.getParameter("userid");
String upws=request.getParameter("userpass");
String logintype = request.getParameter("logintype");
boolean ret = false;
```

```
            TeacherBiz tbiz = new TeacherBizImpl();
            AdminBiz abiz = new AdminBizImpl();
            Teacher t = new Teacher();
            Admin a = new Admin();
            PrintWriter out = response.getWriter();
            if (logintype.equals("teacher") ) {
                        ret = tbiz.isLogin(uid,upws);
                        } else if (logintype.equals("admin")) {
                  ret = abiz.isLogin(uid,upws);
            }
            HttpSession session = request.getSession(true);
            if (ret) {
            if (logintype.equals("teacher"))
                        {t = tbiz.getOneTeacherById(uid);
                        session.setAttribute("user", t);
                  }
                  else if (logintype.equals("admin"))
                        {a = abiz.getOneAdminsById(uid);
                        session.setAttribute("user", a);}
                  session.setAttribute("logintype", logintype);
   out.println("<script>alert('成功登录');window.location.href ='mainpage.
jsp';</script>");
            } else {
                  out.println("<script>alert('输入的账号或密码错误,请重新登录!!
');window.location.href ='login.jsp';</script>");
                  }
```

按照前面导入包的方法，导入相关的包。把 doGet 方法中的内容修改为

```
   doPost(request,response);
```

则完成了 Login 控制器的创建。

任务 4.2　建立 mainpage.jsp 页面

用户登录成功后会跳转到 mainpage.jsp 主页面，在 WebContent 文件夹下建立 mainpage.jsp 页面，其代码如下：

```
   <%@page import="com.ssms.entity.Admin"%>
   <%@page import="com.ssms.entity.Teacher"%>
   <%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>主页面</title>
   </head>
   <body>
      <br><br><br>
```

```
        <h2 align="center">请您选择要做的操作</h2>
        <hr color="red">
        <%      String logintype = session.getAttribute("logintype").toString();
                Teacher t = new Teacher();
                Admin a = new Admin();
                String uid =null;
                String uname =null;
                String pws =null;
                if (!("".equals("logintype") || logintype == null)) {
                        if (logintype.equals("teacher")) {
                                t = (Teacher) session.getAttribute("user");
                                uid =t.getTid();
                                uname = t.getTName();
                                pws = t.getTPassword();
                        } else if (logintype.equals("admin")) {
                                a = (Admin) session.getAttribute("user");
                                uid = a.getAid();
                                uname = a.getaName();
                                pws = a.getaPassword();
                        }
                }
        %>
        <table align="center" border="1" width=160>
                <tr>
                        <td align="center"><a><input type='button'
        onclick="window.location.href('updatepass.jsp?id=<%=uid%>&name=<%=
uname%>&pws=<%=pws%>')"
                                        value=密码修改 style='font-size: 22px'></a>
                </tr>
                <tr></tr><tr></tr>  <tr></tr>
                <tr>
                        <%      if (logintype.equals("teacher")) {
                        %>
                        <td align="center"><input type='submit'
                onclick="window.location.href='TeacherSearchScore'" value="成绩
查询"
                                        style='font-size: 22px' /></td>
                        <%      } else if (logintype.equals("admin")) {
                        %>
                        <td align="center"><input type='submit'
                onclick="window.location.href='AdminSearchScore'" value="成绩查
询"
                                        style='font-size: 22px' /></td>
                        <%      }
                        %>
                </tr>
        </table>
    </body>
    </html>
```

按照模块三中修改配置文件 web.xml 的方法，修改配置文件 web.xml，把欢迎页面更改为 login.jsp，即可运行项目，测试用户登录功能。

任务 5　教师功能的设计与实现

下面完成教师用户对学生成绩的查询、修改、删除和添加功能。

一、建立 ScoreDao 接口

按照前面介绍建立接口的方法，在 com.ssms.dao 包中，建立 ScoreDao 接口，其代码如下：

```
package com.ssms.dao;
import java.util.List;
import com.ssms.entity.NullScore;
import com.ssms.entity.Score;
public interface ScoreDao {
public List<Score> getAllScore();
public Score getOneScoreById(String sno, String cno, String tid);// 按学号、课程号和教师编号查找一条成绩记录
public List<Score> getScoreById(String sno);// 按学号查找成绩
public List<Score> getScoreBySname(String sname);// 按学生姓名查找成绩
public List<Score> getScoreByTid(String tid); // 按教师编号查找成绩
public List<Score> getScoreBySidAndTid(String sid, String tid);
// 按学号和教师编号查找成绩
public List<Score> getScoreBySnameAndTid(String sname, String tid);
// 按学生姓名和教师编号查找成绩
public List<Score> getScoreByTotalScorePrecise(int totalscore, String tid);
// 按教师编号进行成绩精确查找
public List<Score> getScoreByTotalScorePrecise(int totalscore);// 成绩精确查找
public List<Score> getScoreByTotalScoreCompare(String totalscore, String tid);
// 按教师编号进行成绩比较查找
public List<Score> getScoreByTotalScoreCompare(String totalscore);// 成绩比较查找
public List<Score> getScoreByTotalScoreSet(String totalscore, String tid);
//按教师编号进行成绩集合查找
public List<Score> getScoreByTotalScoreSet(String totalscore);//成绩集合查找
public List<Score> getScoreByTotalScoreScope(String tid, int begin, int end);
//按教师编号进行成绩范围查找
public List<Score> getScoreByTotalScoreScope(int begin, int end);// 成绩范围查找
public List<NullScore> getScoreByTotalScoreNull(String tid);
// 按教师编号进行成绩空值查找
public List<NullScore> getScoreByTotalScoreNull();// 成绩空值查找
public int updateScore(String sno, String cname, int usualscore, int experimentscore, int examscore,  int totalscore);//更新成绩
```

```
    public boolean isDeleteScore(String sid, String cid, String tid);//删除成
绩
    public boolean isAddScore(String sid, String cid, String tid, int usualScore,
int experimentScore, int examScore,            int totalScore);//添加成绩
    public boolean isBackupScore(String path);//成绩备份
    public boolean isRestoreScore(String path);//成绩恢复
    }
```

二、建立 ScoreDao 接口的实现类 ScoreDaoImpl

按照前面建立类的方法和步骤在 com.ssms.dao.impl 包中建立 ScoreDaoImpl 类，实现 ScoreDao 接口。为了提高程序运行效率，对数据库中数据的处理采用存储过程实现。该类中查找所有课程的方法 getAllScore()中调用存储过程 proc_select_allscore，该存储过程的作用是查找所有学生的成绩信息，创建存储过程的代码是：

```
    use  ssms
    go
    if(object_id('proc_select_allscore', 'p') is not null)
    drop proc proc_select_allscore
    go
    create proc proc_select_allscore
    as
    begin
    select  *   from v_stu_score  --其中的 v_stu_score 是在模块三中建立的视图
    end
```

调用存储过程代码是：

```
    pstmt = conn.prepareStatement("{call dbo.proc_select_allscore}");
```

在成绩范围查找方法 getScoreByTotalScoreScope 中，调用存储过程 proc_get_score_scope，该存储过程的作用是一个带输入参数的存储过程，其功能是按给定教师编号和成绩的范围区间查找所有学生的成绩信息，创建存储过程的代码是：

```
    use  ssms
    go
    if(OBJECT_ID('proc_get_score_scope', 'P') is not null)
    drop proc proc_get_score_scope
    go
    create proc proc_get_score_scope(@tid varchar(8),@begin int, @end int)
    as
    select  *   from v_stu_score where  tid=@tid and totalScore  between  @begin
and @end
    go
```

调用该存储过程的代码是：

```
    pstmt = conn.prepareStatement("{call dbo.proc_get_score_scope(?,?,?)}");
```

存储过程有多种形式的，比如带输入输出参数的、有返回值的、带游标的存储过程等，

由于本书用到系统中没有用到这些类型的存储过程，在这里不再介绍其具体使用，在后面相关知识部分会对其他形式的存储过程进行详细介绍，如果想了解其他形式存储过程的定义和调用，请参阅相关知识。系统其他部分对数据库中数据的访问，没有采用存储过程实现，采用参数化 SQL 语句实现，作为练习请读者把系统中所有对数据的操作都改成存储过程来实现。

ScoreDaoImpl 类的代码如下：

```
   package com.ssms.dao.impl;
   import java.sql.Connection;
   import java.sql.PreparedStatement;
   import java.sql.ResultSet;
   import java.sql.SQLException;
   import java.util.ArrayList;
   import java.util.List;
   import javax.swing.JOptionPane;
   import com.ssms.dao.BaseDao;
   import com.ssms.dao.ScoreDao;
   import com.ssms.entity.NullScore;
   import com.ssms.entity.Score;
   public class ScoreDaoImpl  implements  ScoreDao {
      public List<Score> getAllScore() {
             Connection conn = null;
             PreparedStatement pstmt = null;
             ResultSet rs = null;
             List<Score> list = new ArrayList<Score>();
             conn = BaseDao.getConn();
             try  {pstmt  =  conn.prepareStatement("{call  dbo.proc_select_
allscore}");
                   rs = pstmt.executeQuery();
                   while (rs.next()) {
                          Score score = new Score();
                          score.setSid(rs.getString("sid"));
                          score.setSname(rs.getString("sname"));
                          score.setCname(rs.getString("cname"));
                          score.setsClassName(rs.getString("sClassName"));
                          score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                          score.setExamResults(rs.getInt("examResults"));
                          score.setTotalScore(rs.getInt("totalScore"));
                          list.add(score);
                   }
             } catch (Exception e) {
                   e.printStackTrace();
             } finally {  BaseDao.closeAll(rs, pstmt, conn);
             }
             return list;
      }
   public Score getOneScoreById(String sno, String cno, String tid) {
```

```
            Score score = new Score();
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            conn = BaseDao.getConn();
            String sql = "select  *   from v_stu_score  where student.sid=?
and  course.cid=? and  tid=?";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, sno);
                  pstmt.setString(2, cno);
                  pstmt.setString(3, tid);
                  rs = pstmt.executeQuery();
                  if (rs.next()) {    score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setCid(rs.getString("cid"));
                        score.setTid(rs.getString("tid"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                  }
            } catch (SQLException e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return score;
      }
      public List<Score> getScoreByTid(String tid) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  tid=?";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, tid);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCid(rs.getString("cid"));
                        score.setCname(rs.getString("cname"));
                        score.setTid(rs.getString("tid"));
                        score.setsClassName(rs.getString("sClassName"));
```

```
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreBySidAndTid(String sid, String tid) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  student.sid=?
and tid=?";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, sid);
                  pstmt.setString(2, tid);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreBySnameAndTid(String sname, String tid) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
```

```
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  tid=? and sname
like '%"+ sname + "%' ";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, tid);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreByTotalScorePrecise(int totalscore, String
tid) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  tid=?  and
totalScore=?";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, tid);
                  pstmt.setInt(2, totalscore);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
```

```
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreByTotalScoreCompare(String totalscore,
String tid) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            int pmax = totalscore.indexOf(">");
            int pmaxequ = totalscore.indexOf(">=");
            int pmin = totalscore.indexOf("<");
            int pminequ = totalscore.indexOf("<=");
            int totalscoreint = 0;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            String sql = "";
            conn = BaseDao.getConn();
            if (pmax != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(1));
      sql = " select * from v_stu_score where tid=? and totalScore >"+
totalscoreint;
            } else if (pmaxequ != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(2));
      sql = " select * from v_stu_score where tid=? and totalScore >="+
totalscoreint;
            } else if (pmin != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(1));
   sql = " select * from v_stu_score where tid=? and totalScore <"+
totalscoreint;
            } else if (pminequ != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(2));
   sql = " select * from v_stu_score where tid=? and totalScore <="+
totalscoreint;
            }
            try { pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, tid);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));
```

```
      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      @Override
     public List<Score> getScoreByTotalScoreSet(String totalscore, String tid)
{
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
   String sql = " select  *  from v_stu_score  where  tid=? and totalScore  in
("     + totalscore + ")";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, tid);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
   public List<Score> getScoreByTotalScoreScope(String tid, int begin, int end)
{
            Connection conn = null;
            PreparedStatement pstmt = null;
```

```
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
   try          {pstmt          =          conn.prepareStatement("{call
dbo.proc_get_score_scope(?,?,?)}");
                  pstmt.setString(1, tid);
                  pstmt.setInt(2, begin);
                  pstmt.setInt(3, end);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<NullScore> getScoreByTotalScoreNull(String tid) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<NullScore> list = new ArrayList<NullScore>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  tid=?  and
totalScore is null";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, tid);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        NullScore score = new NullScore();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults("null");
                        score.setExperimentalResults("null");
                        score.setExamResults("null");
                        score.setTotalScore("null");
```

```
                        list.add(score);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                    BaseDao.closeAll(rs, pstmt, conn);
                }
                return list;
        }
        public int updateScore(String sno, String cname, int usualscore, int
experimentscore, int examscore,  int totalscore) {
                Connection conn = null;
                PreparedStatement pstmt = null;
                ResultSet rs = null;
                conn = BaseDao.getConn();
                int i = 0;
                String sql = "update score set usualResults=?,examResults=?,
experimentalResults=?,totalScore=? where sid=? and cid =(select cid from
course where cname=?)";
                try {  pstmt = conn.prepareStatement(sql);
                    pstmt.setInt(1, usualscore);
                    pstmt.setInt(2, examscore);
                    pstmt.setInt(3, experimentscore);
                    pstmt.setInt(4, totalscore);
                    pstmt.setString(5, sno);
                    pstmt.setString(6, cname);
                    i = pstmt.executeUpdate();
                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                    BaseDao.closeAll(rs, pstmt, conn);
                }
                return i;
        }
        public boolean isDeleteScore(String sid, String cid, String tid) {
                boolean ret = false;
                Connection conn = null;
                PreparedStatement pstmt = null;
                conn = BaseDao.getConn();
                String sql = "delete from score where sid=? and cid=? and tid=? ";
                try {  pstmt = conn.prepareStatement(sql);
                    pstmt.setString(1, sid);
                    pstmt.setString(2, cid);
                    pstmt.setString(3, tid);
                    int num = pstmt.executeUpdate();
                    if (num != 0) {
                        ret = true;
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
```

```
            } finally {
                  BaseDao.closeAll(null, pstmt, conn);
            }
            return ret;
      }
      public boolean isAddScore(String sid, String cid, String tid, int
usualScore, int experimentScore, int examScore,
                  int totalScore) {
            boolean ret = false;
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            conn = BaseDao.getConn();
   String sql = "select * from score where sid= '" + sid + "' and cid='" + cid
+ "' and tid='" + tid + "'";
            try {  pstmt = conn.prepareStatement(sql);
                  rs = pstmt.executeQuery();
                  if (rs.next()) {
      JOptionPane.showMessageDialog(null, "已经存在该生该课程的成绩！");
                        return ret;
                  }
                  String sqlinsert = "insert into score (sid,cid,tid,
usualResults,examResults,experimentalResults,totalScore)
values(?,?,?,?,?,?,?)";
                  pstmt = conn.prepareStatement(sqlinsert);
                  pstmt.setString(1, sid);
                  pstmt.setString(2, cid);
                  pstmt.setString(3, tid);
                  pstmt.setInt(4, usualScore);
                  pstmt.setInt(5, experimentScore);
                  pstmt.setInt(6, examScore);
                  pstmt.setInt(7, totalScore);
                  int num = pstmt.executeUpdate();
                  if (num != 0) {
                        ret = true;
                  }
            } catch (SQLException e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(null, pstmt, conn);
            }
            return ret;
      }
      public List<Score> getScoreById(String sno) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  student.
```

```
sid=?";
              try {
                     pstmt = conn.prepareStatement(sql);
                     pstmt.setString(1, sno);
                     rs = pstmt.executeQuery();
                     while (rs.next()) {
                            Score score = new Score();
                            score.setSid(rs.getString("sid"));
                            score.setSname(rs.getString("sname"));
                            score.setCname(rs.getString("cname"));
                            score.setsClassName(rs.getString("sClassName"));
                            score.setUsualResults(rs.getInt("usualResults"));

       score.setExperimentalResults(rs.getInt("experimentalResults"));
                            score.setExamResults(rs.getInt("examResults"));
                            score.setTotalScore(rs.getInt("totalScore"));
                            list.add(score);
                     }
              } catch (Exception e) {
                     e.printStackTrace();
              } finally {
                     BaseDao.closeAll(rs, pstmt, conn);
              }
              return list;
       }
       public List<Score> getScoreBySname(String sname) {
              Connection conn = null;
              PreparedStatement pstmt = null;
              ResultSet rs = null;
              List<Score> list = new ArrayList<Score>();
              conn = BaseDao.getConn();
       String sql = " select  *  from v_stu_score  where  sname like '%"  + sname
+ "%' ";
              try {  pstmt = conn.prepareStatement(sql);
                     rs = pstmt.executeQuery();
                     while (rs.next()) {
                            Score score = new Score();
                            score.setSid(rs.getString("sid"));
                            score.setSname(rs.getString("sname"));
                            score.setCname(rs.getString("cname"));
                            score.setsClassName(rs.getString("sClassName"));
                            score.setUsualResults(rs.getInt("usualResults"));

       score.setExperimentalResults(rs.getInt("experimentalResults"));
                            score.setExamResults(rs.getInt("examResults"));
                            score.setTotalScore(rs.getInt("totalScore"));
                            list.add(score);
                     }
              } catch (Exception e) {
                     e.printStackTrace();
```

```
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreByTotalScorePrecise(int totalscore) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  totalScore=?";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setInt(1, totalscore);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreByTotalScoreCompare(String totalscore) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            int pmax = totalscore.indexOf(">");
            int pmaxequ = totalscore.indexOf(">=");
            int pmin = totalscore.indexOf("<");
            int pminequ = totalscore.indexOf("<=");
            int totalscoreint = 0;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            String sql = "";
            conn = BaseDao.getConn();
            if (pmax != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(1));
  sql = " select  *   from v_stu_score  where  totalScore >"       +
```

```
totalscoreint;
            } else if (pmaxequ != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(2));
   sql = " select  *  from v_stu_score  where  totalScore >="   + totalscoreint;
            } else if (pmin != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(1));
   sql = " select  *   from v_stu_score  where  totalScore <"+ totalscoreint;
            } else if (pminequ != -1) {
                  totalscoreint = Integer.valueOf(totalscore.substring(2));
                  sql = select  *  from v_stu_score  where  totalScore <=" +
totalscoreint;
            }
            try {  pstmt = conn.prepareStatement(sql);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreByTotalScoreSet(String totalscore) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
   String sql = " select  * from v_stu_score  where  totalScore  in ("    +
totalscore + ")";
            try {  pstmt = conn.prepareStatement(sql);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
```

```
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<Score> getScoreByTotalScoreScope(int begin, int end) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            List<Score> list = new ArrayList<Score>();
            conn = BaseDao.getConn();
            String sql = " select *  from v_stu_score  where totalScore
between  ?  and ? ";
            try {  pstmt = conn.prepareStatement(sql);
                  pstmt.setInt(1, begin);
                  pstmt.setInt(2, end);
                  rs = pstmt.executeQuery();
                  while (rs.next()) {
                        Score score = new Score();
                        score.setSid(rs.getString("sid"));
                        score.setSname(rs.getString("sname"));
                        score.setCname(rs.getString("cname"));
                        score.setsClassName(rs.getString("sClassName"));
                        score.setUsualResults(rs.getInt("usualResults"));

      score.setExperimentalResults(rs.getInt("experimentalResults"));
                        score.setExamResults(rs.getInt("examResults"));
                        score.setTotalScore(rs.getInt("totalScore"));
                        list.add(score);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            } finally {
                  BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public List<NullScore> getScoreByTotalScoreNull() {
            Connection conn = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
```

```
            List<NullScore> list = new ArrayList<NullScore>();
            conn = BaseDao.getConn();
            String sql = " select  *   from v_stu_score  where  totalScore is
null";
            try {  pstmt = conn.prepareStatement(sql);
                   rs = pstmt.executeQuery();
                   while (rs.next()) {
                          NullScore score = new NullScore();
                          score.setSid(rs.getString("sid"));
                          score.setSname(rs.getString("sname"));
                          score.setCname(rs.getString("cname"));
                          score.setsClassName(rs.getString("sClassName"));
                          score.setUsualResults("null");
                          score.setExperimentalResults("null");
                          score.setExamResults("null");
                          score.setTotalScore("null");
                          list.add(score);
                   }
            } catch (Exception e) {
                   e.printStackTrace();
            } finally {
                   BaseDao.closeAll(rs, pstmt, conn);
            }
            return list;
      }
      public boolean isBackupScore(String path) {
            boolean ret = false;
            Connection conn = null;
            PreparedStatement pstmt = null;
            conn = BaseDao.getConn();
            try {  String sql = "backup  database ssms to disk= '" + path + "'
with init";
                   pstmt = conn.prepareStatement(sql);
                   int num =0;
                num=pstmt.executeUpdate();
                   if (num != 0) {
                          ret = true;
                   }
            } catch (SQLException e) {
                   e.printStackTrace();
            } finally {
                   BaseDao.closeAll(null, pstmt, conn);
            }
            return ret;
      }
      public boolean isRestoreScore(String path) {
            boolean ret = false;
            Connection conn = null;
            PreparedStatement pstmt = null;
            conn = BaseDao.getConn();
```

```
    try {     String sql = "alter database ssms set offline with rollback
immediate;use master;"
                                + "restore database ssms from disk= '" +
path.trim()
                                + "' with replace;alter database ssms set
online with rollback immediate";
                    pstmt = conn.prepareStatement(sql);
                    int num = -1;
                    num = pstmt.executeUpdate();
                    if (num == 0) {
                          ret = true;
                    }
             } catch (SQLException e) {
                    e.printStackTrace();
             } finally {
                    BaseDao.closeAll(null, pstmt, conn);
             }
             return ret;
       }
    }
```

相关知识

一、游标

一般情况下，使用 select 语句为变量赋值时，只能将结果集的一条记录赋值给一组变量，但是，在项目开发中，往往需要同时处理一条或者多条记录，单纯的变量赋值很难满足开发需要，游标提供了一种对从表中检索出的数据进行操作的灵活手段，就本质而言，游标实际上是一种能从包括多条数据记录的结果集中每次提取一条记录的机制。游标可以理解成一个定义在特定数据集上的指针，通过控制这个指针遍历数据集，或者仅仅是指向特定的行，所以游标是定义在以 select 开始的数据集上，实现对 select 语句返回的结果集进行逐条处理。

游标的类型：

（1） 静态游标：该游标不检测数据行的变化；

（2） 动态游标：该游标反映所有数据行的改变；

（3） 仅向前游标：该游标不支持滚动；

（4） 键集游标：该游标能反映修改操作，但不能准确反映插入、删除操作。

游标使用顺序：

（1） 定义游标；

（2） 打开游标；

（3） 通过循环从游标中检索记录；

（4） 关闭游标；

（5） 释放游标。

1. 声明游标的语法格式

```
declare 游标名 cursor
[local | global]
[forward_only | scroll]
[static | keyset | dynamic |fast_forward]
[read_only | scroll_locks]
```

for selet 语句

```
 [for  update[of 列名[，列名]]
```

格式说明：

（1） local：约定游标是局部游标，定义游标的作用域仅限其所在的批处理、存储过程或触发器中，当游标所在的批处理、存储过程或触发器执行结束后，游标会被自动释放；global 约定游标是全局游标，即该游标在由连接执行的任何存储过程或批处理中都可以引用该游标名称，该游标仅在断开连接时隐性释放。如果不指定，则默认为 global。

（2）forward_only：约定游标只能从第一行滚动到最后一行，即 fetch 只适用于 fetch next 形式；scroll 约定游标读取数据集数据时，可以根据需要，向任何方向或位置移动。若不指定，则默认为 forward_only。

（3） static：用于定义一个静态游标，即游标被定义后，在系统临时数据库 tempdb 中创建该游标使用的数据临时副本。对游标的所有请求都从临时数据库 tempdb 中的临时表中得到应答。因此，在对该游标进行读取操作时所返回的数据，并不反映对基本表所做的修改（此后对基本表内的数据修改，将不会影响游标中数据集的数据），并且该游标不允许修改。dynamic 与 static 完全相反，当基本表中数据库修改时，游标的内容也随之得到反应，在下一次 fetch 中，数据内容会随之改变，但动态游标不支持 absolute 选项。对比 static 与 dynamic 发现二者的区别：static 游标一经定义，其结果集不随基本表数据的修改而修改，而 dynamic 定义的游标结果集随基本表数据的修改而修改。

（4） keyset：指定当游标打开时，游标中行的成员资格和顺序已经固定。对行进行唯一标识的键集内置在临时数据库 tempdb 内 keyset 表中。对基表中的非键值所做的更改（由游标所有者更改或由其他用户提交）在用户滚动游标时是可视的。其他用户进行的插入是不可视的（不能通过 Transact-sql 服务器游标进行插入）。如果某行已删除，则对该行的提取操作将返回 @@fetch_status 值-2。从游标外更新键值类似于删除旧行后接着插入新行的操作。含有新值的行不可视，对含有旧值的行的提取操作将返回 @@fetch_status 值-2。如果通过指定 where current of 子句用游标完成更新，则新值可视。

（5） read_only：指定声明的游标只能读取数据，而不能做任何更新操作。

（6） scroll_locks：指定确保通过游标完成的定位更新或定位删除可以成功。当将行读入游标以确保它们可用于以后的修改时，SQL Server 会锁定这些行。如果还指定了 fast_forward，则不能指定 scroll_locks。

（7） fast_forward：可以理解成 forward_only 的优化版本。forward_only 执行的是表态计划，而 fast_forward 是根据情况进行选择采用动态计划还是静态计划，大多数情况下

fast_forward 要比 forward_only 性能略好。指定了该选项，则不可指定 scroll 或 for_update。

（8） optimistic：选项并不会对游标查询数据集进行研究锁定，但游标基本表数据发生了更新，则通过游标进行更新操作将不会被执行，即游标操作失败；若基本表数据未发生更新，则通过游标进行的操作继续被执行。

2. 打开游标

打开游标的语法格式：

```
open [global] 游标名称
```

格式说明：

global 表示需要打开的是全局游标。

判断打开游标是否成功，可通过全局变量@@errorr 的返回值来确定。如果@@error 等于 0，表示打开成功，否则表示打开失败。当声明的游标是表态游标或 keyset 游标时，可通过全局变量@@cursor_rows 的返回值获取游标中的记录数。

@@cursor_rows，表示游标中结果集中的行数，其值亦有四种：

n（游标中记录数）；

−1（游标是动态的）；

0（空集游标）；

−m（正在从基本表向游标读入数据，m 表示当前游标中的数据行数）。

3. 获取游标中数据

当一个游标成功打开之后，就可以使用 fetch 语句读取游标中的数据。

```
fetch [[next | prior | first | last |
absolute{ n | @nvar | relative { n | @nvar}]
from [global ] 游标名 [into 变量]
```

格式说明：

（1） next：返回当前记录的下一条记录，并移动记录指针到当前位置。如果 fetch next 为对游标的第一次提取操作，则返回结果集中的第一条记录。NEXT 为默认的游标提取选项。

（2） prior：返回当前记录的上一条记录，并移动记录指针到当前位置。如果 FETCH PRIOR 为对游标的第一次提取操作，则没有记录返回，并且记录指针置于第一条记录之前。

（3） first：返回游标中的第一条记录并将其作为当前记录。

（4） last：返回游标中的最后一条记录并将其作为当前记录。

（5） ABSOLUTE n | @nvar：绝对位置提取。如果 n | @nvar 是大于零的数，则返回从游标头部开始的第 N 条记录，并将返回的记录作为当前记录；如果 n | @nvar 是小于零的数，则返回从游标尾部开始的第 N 条记录，同样将返回的记录作为当前记录；如果 n | @nvar 等于零，则没有记录返回，因此，当前记录不变。

（6） relative n | @nvar：相对位置提取。与 absolute 不同之处在于 relative 是从当前记录位置向前（n | @nvar 的值小于零）或者向后（n | @nvar 的值大于零）提取记录，同时，将返回的记录作为当前记录。

（7） into 变量：将提取操作的列数据放到局部变量中。列表中的各个变量从左到右与游标结果集中的相应列相关联。各变量的数据类型必须与相应的结果列的数据类型匹配或者结果列数据类型所支持的隐性转换。变量的数目必须与游标选择列表中的列的数目一致。

要判断游标数据提取是否成功，可通过游标状态变量@@fetch_status 的返回值来确定，即每次用 fetch 语句从游标中提取数据时，都应检查该变量，以确定上次 fetch 操作是否成功，以确定下一步的处理。

游标状态变量@@fetch_status 取值有三种：

0（提取成功）；

-1（提取失败）；

-2（需要提取的记录不存在）。

4. 关闭游标

在处理完游标中数据之后，使用 CLOSE 命令关闭游标。一个已打开的游标，其语法如下：

语法格式：

```
close [golobal] 游标名
```

close 语句仅关闭游标，但不释放游标占用的数据结构，关闭后的游标在需要时仍然可以使用 open 命令再次打开。

5. 释放游标

当确定某个游标不再使用时，应当及时使用 deallocate 命令释放游标，SQL Server 2008 将删除这个游标的数据结构。

语法格式：

```
deallocate [global] 游标名
```

6. 游标的应用

应用 1　利用游标修改、删除数据

```
declare cur_stdent1 cursor scroll for
select sid,sName,sSex,sBirthday
from student for update of sBirthday
open cur_stdent1
declare @rowcount int
set @rowcount=3
fetch absolute @rowcount from cur_stdent1
--将变量@rowcount 标识的行设置为当前行
--下面行是利用游标进行修改操作
update student set sBirthday='1991-08-16' where current of cur_stdent1 --
修改游标中的当前行
--下面行是利用游标进行删除操作
delete from student where current of cur_stdent1
```

```
close cur_stdent1
deallocate cur_stdent1
```

应用 2　利用游标对查询的数据集进行遍历

```
use ssms;
declare cur_student2 cursor local for
select sid,sName,sSex,2016-year(sBirthday) as Sage from student
open cur_student1;
declare @sno char(100),@name varchar(10)
declare @sex char(2),@age int
fetch next from cur_student2 into @sno,@name,@sex,@age
while(@@FETCH_STATUS=0)
begin
print '学号：'+@sno
print '姓名：'+@name
print '性别：'+@sex
print '年龄：'+convert(char,@age)
print '----------------------'
fetch next from cur_student2 into @sno,@name,@sex,@age
end
close cur_student2;
deallocate cur_student2;
```

对于游标 cur_student2，默认是 forward_only 型，由于该游标是 local 类型，则运行时必须将声明游标与从游标中提取值一起运行，否则先执行声明再执行从游标中提取数据时提示错误信息“名为'cur_student2'的游标不存在”，原因是局部游标批处理结束后已经被系统隐式释放，无法再被其他批处理调用。若需要单独运行，则应修改为 global 类型。

应用 3　利用游标对查询的数据集进行随意提取

```
declare cur_score1 cursor global scroll for
select student.sid,sName,course.cid,cName,totalScore as grade
from student,course,score
where student.sid=score.sid and course.cid=score.cid
order by grade
open cur_score1
declare @sno char(10),@cno varchar(10),@grade int
declare @sname varchar(10),@cname varchar(10)
declare @temp int
fetch first from cur_score1 into @sno,@sname,@cno,@cname,@grade
set @temp=@grade
while(@@FETCH_STATUS=0 and @temp=@grade)
begin
print '最低分是：'
print '学号：'+@sno
print '姓名：'+@sname
print '课程号：'+@cno
```

```
print '课程名：'+@cname
print '成绩：'+convert(char,@grade)
print '-----------------'
set @temp=@grade
fetch next from cur_score1 into @sno,@sname,@cno,@cname,@grade
end
fetch last from cur_score1 into @sno,@sname,@cno,@cname,@grade
set @temp=@grade
while(@@FETCH_STATUS=0 and @temp=@grade)
begin
print '最高分是：'
print '学号：'+@sno
print '姓名：'+@sname
print '课程号：'+@cno
print '课程名：'+@cname
print '成绩：'+convert(char,@grade)
print '-----------------'
set @temp=@grade
fetch relative -1 from cur_score1 into @sno,@sname,@cno,@cname,@grade
end
close cur_score1
deallocate cur_score1
```

关于游标一些建议：

（1） 如果能不用游标，尽量不要使用游标；

（2） 用完之后一定要及时关闭并释放游标；

（3） 尽量不要在大量数据上定义游标；

（4） 尽量不要使用游标更新数据；

（5） 尽量不要使用 insensitive、static 和 keyset 这些参数定义游标；

（6） 如果可以，尽量使用 fast_forward 关键字定义游标；

（7） 如果只对数据进行读取，当读取时只用到 fetch next 选项，则最好使用 forward_only 参数。

二、存储过程

存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的 SQL 语句集合，经编译后存储在数据库中，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。

在性能方面，存储过程有如下的优点：

（1） 预编译。存储过程预先编译好放在数据库内，减少编译语句所花的时间。

（2） 缓存。编译好的存储过程会进入缓存，所以对于经常执行的存储过程，除了第一次执行外，其他次执行的速度会有明显提高。

（3） 减少网络流量。减少网络数据传输量，特别对于处理一些数据的存储过程，不必和 SQL 语句实现那样多次传送数据到客户端。

（4） 安全性提高。参数化的存储过程可以防止 SQL 注入式攻击。

1. 存储过程分类

存储过程可以分为系统存储过程和用户自定义存储过程两种。

（1） 系统存储过程。

系统存储过程是系统创建的存储过程，目的在于能够方便地从系统表中查询信息或完成与更新数据库表相关的管理任务或其他的系统管理任务。系统存储过程主要存储在 master 数据库中，以“sp”下划线开头的存储过程。尽管这些系统存储过程在 master 数据库中，但在其他数据库可以直接调用这些系统存储过程，另外，有一些系统存储过程会在创建新的数据库的时候被自动创建在当前数据库中。

常用系统存储过程有：

```
use ssms;
exec sp_columns student;--查看学生表中列的信息
exec sp_helpIndex student;--查看学生表中创建的索引信息
exec sp_helpConstraint student;--查看学生表中创建的约束信息
exec sp_stored_procedures; --查看当前数据库中所有存储过程的信息
exec sp_helptext 'sp_stored_procedures';--查看存储过程 sp_stored_procedures
创建、定义语句
exec sp_rename student, stuInfo;
--对表 student 重命名为 stuInfo，该存储过程也可以对索引、列重命名
exec sp_renamedb ssms, ss;--对数据库 ssms 重命名为 ss
exec sp_helpdb;--数据库帮助，查询数据库信息
exec sp_helpdb master;--数据库 master 详细信息查询用户自定义存储过程
```

（2） 用户自定义存储过程。

2. 创建存储过程

创建语法格式。

```
create proc | procedure pro_name
[{@参数  参数类型[varying]} [=default] [output][1…]
[with encryption | recompile]
   for replication]
as
   SQL_statements
```

说明：

① pro_name：用户自定义存储过程的名称，在命名时通常遵守“见名知义”的规则。

② @参数：存储过程的参数。声明存储过程的参数与声明局部变量的参数一样，必须以“@”开头，后面跟参数名称，参数名称必须符合标识符的规则。用户自定义的存储过程可以没有参数，也可以有一个或多个参数。用户必须在执行存储过程时提供每个所声明参数的值（除非定义了该参数的默认值）。每个存储过程的参数仅用于该存储过程本身，因此，不同的存储可以有相同的参数名称。默认情况下，参数只能代替常量，而不能用于代替表名、列名或其他数据库对象的名称。

③ 参数类型：参数的数据类型。SQL Server 提供的所有数据类型（包括 text、ntext 和 image）均可以用作存储过程中参数的类型。但是，cursor 数据类型只能用于 输出参数。

如果指定的数据类型为 cursor，也必须同时指定 varying 和 output 关键字。有关 SQL Server 提供的数据类型及其语法的更多信息，请参考相关资料。

④ varying：指定游标作为输出参数。

⑤ default：参数的默认值。如果定义了默认值，不必指定该参数的值即可执行过程。默认值必须是常量或 NULL。如果过程将对该参数使用 LIKE 关键字，那么默认值中可以包含通配符（%、_、[]和[^]）。

⑥ output：表明参数是输出参数。该选项的值可以返回给 exec[ute]。使用 output 参数可将信息返回给调用过程。text、ntext 和 image 参数可用作 output 参数。使用 output 关键字的输出参数可以是游标占位符。

⑦ recompile：表示每次运行存储过程时，都对存储过程进行重新编译。在使用非典型值或临时值而不希望覆盖缓存在内存中的执行计划时，请使用 recompile 选项。

⑧ encryption：表示 SQL Server 加密 syscomments 表中包含 create procedure 语句文本的条目。使用 encryption 可防止将存储过程作为 SQL Server 复制的一部分发布。说明在升级过程中，SQL Server 利用存储在 syscomments 中的加密注释来重新创建加密过程。

⑨ for replication：指定不能在订阅服务器上执行为复制创建的存储过程。使用该选项创建的存储过程可用作存储过程筛选器，且只能在复制过程中执行。本选项不能和 with recompile 选项一起使用。

⑩ as：指定存储过程要执行的操作。

⑪ sql_statement：存储过程中要包含的任意数目和类型的 Transact-SQL 语句，但有一些限制。

3. 执行存储过程

```
exec|execute  procedure_name
```

执行指定的存储过程，存储过程如果有参数，后面加参数格式为：@参数名=value，也可直接给出参数值 value。

4. 修改存储过程

```
alter proc | procedure procedure_name
as
SQL_statements
```

修改指定的存储过程为对应的 Transact-SQL 语句。

5. 存储过程的删除

```
drop proc | procedure procedure_name
```

删除指定的存储过程。

注意：在存储过程中能调用另外一个存储过程，而不能删除另外一个存储过程。

6. 存储过程的应用实例

应用 1　只返回单一记录集的存储过程

【例 1】创建存储过程 proc_get_student，如果该存储过程已经存在，则先删除，后创建。

```
if(exists (select * from sys.objects where name = 'proc_get_student'))
drop proc proc_get_student
go
create proc proc_get_student
as
select * from student;
--调用、执行存储过程
exec proc_get_student;
```

结果：相当于运行“select * from student”这行代码，结果为整个表的数据。

```
--修改存储过程
alter proc proc_get_student
as
select sid,sName,sClassName,sBirthday from student;
```

【例 2】创建加密存储过程 pro_get_sage 用于给出学生年龄。

```
go
if(OBJECT_ID('view_age','v') is not null)
drop view view_age
go
create view view_age
as
select sid,sName, YEAR (getdate())-YEAR(sBirthday) as sAge,sSex from
student;
go
if(OBJECT_ID('pro_get_sage','p') is not null)
drop proc pro_get_sage
go
create proc pro_get_sage
with encryption
as
select * from view_age
go
exec pro_get_sage
exec sp_helptext 'pro_get_sage';
exec sp_helptext 'proc_get_student';
```

说明：调用 object_id 函数判断存储过程 proc_get_score 是否存在，该函数要求两个参数，第一个参数是存储过程的名字，第二个参数是对象类型为“P”，即存储过程。

结果：相当于运行“select sid,sName,2016-YEAR(sBirthday) as sAge,sSex from student”这行代码。

应用 2　没有输入输出参数的存储过程

【例 1】创建不缓存存储过程 pro_insert_student 实现将学号为“1334110130”，初始密码为“000000”，姓名“杨阳”的学生信息插入到学生表 student。

```
if(OBJECT_ID('pro_insert_student','p') is not null)
drop proc pro_insert_student
go
create Proc pro_insert_student
with recompile
as
insert into student(sid,sPassword,sName) values('1334110130','000000','杨阳')
go
exec pro_insert_student;
```

结果：相当于运行“insert into student(sid,sPassword,sName) values('1334110130','000000','杨阳')”这行代码，结果为插入指定数据到 student 表。

【例 2】创建存储过程 pro_update_student 实现将学号是“1334110130”的姓名修改为“杨婷”。

```
if(OBJECT_ID('pro_update_student','p') is not null)
drop proc pro_update_student
go
create proc pro_update_student
as
update student set sName='杨婷' where sid='1334110130'
go
exec pro_update_student;
```

结果：相当于运行“update student set sName='杨婷' where sid='1334110130'”这行代码，结果为修改学生表 student 中的数据。

应用 3　有返回值的存储过程

【例 1】创建存储过程 pro_update_score 实现当对成绩表 score 执行修改操作时返回受影响的行数。

```
if(OBJECT_ID('pro_update_score','p') is not null)
drop proc pro_update_score
go
create proc pro_update_score
as
declare @count int
update score set totalScore=totalScore+1 where cid='4'
set @count=@@ROWCOUNT
return @count
go
declare @c int;
exec @c=pro_update_score
```

```
    select @c as '此操作影响行数'
```

说明：这里的@@ROWCOUNT 为返回受上一语句影响的行数。

应用 4　有输入参数和输出参数的存储过程

【例 1】创建存储过程 proc_get_score 实现查询指定区间的成绩信息。

```
    if(OBJECT_ID('proc_get_score', 'P') is not null)
    drop proc proc_get_score
    go
    create proc proc_get_score(@min int, @max int)
    as
    select * from score where totalScore between @min and @max
    go
    exec proc_get_score 80,100;
```

结果：相当于运行“select * from score where totalScore between 80 and 100”这行代码，结果为整个表的数据。

【例 2】创建存储过程 proc_get_StudentByName 实现按姓名模糊查询功能。

```
    if(object_id('proc_get_StudentByName', 'p') is not null)
    drop proc proc_get_StudentByName
    go
    create proc proc_get_StudentByName(@name varchar(20) = '%阳%', @nextName
varchar(20) = '%')
    as
    select * from student where sName like @name and sName like @nextName;
    go
    exec proc_get_StudentByName;--利用默认值姓名中包括“阳”字查询学生信息
    exec proc_get_StudentByName '%冰%', '刘%';
    --利用传入参数查询姓名中包括有“冰”字并且姓“刘”的学生信息。
```

【例 3】创建存储过程 proc_get_avgscore_id 输入学生学号，查询该生的平均分。

```
    if(object_id('proc_get_avgscore_id', 'p') is not null)
    drop proc proc_get_avgscore_id
    go
    create proc proc_get_avgscore_id
    (
    @sid char(10),
    @avgscore int output
    )
    as
    select @avgscore=AVG(totalscore)from score where sid=@sid;
    go
    declare @sid  char(10),@avgscore int
    set @sid='1234120101'
    exec proc_get_avgscore_id @sid,@avgscore output
    select @sid '学号',@avgscore '平均分'
    --不正确的调用形式
```

```
declare @sid  char(10),@avgscore int
set @sid='1234120101'
exec proc_get_avgscore_id @sid,@avgscore
select @sid '学号',@avgscore '平均分'
```

注意带输出参数的存储过程调用的时候，输出参数也要加 output。

带有输入输出参数的存储过程在 JSP 中的调用要用到 CallableStatement 对象，CallableStatement 继承 Statement 的方法（Statement 用于处理一般的 SQL 语句），还继承了 PreparedStatement 的方法（PreparedStatement 用于处理 IN 参数）。CallableStatement 中定义的所有方法都用于处理 out 参数或 inout 参数的输出部分。

下面举例说明如何在 JSP 中调用带输入、输出参数的存储过程。具体代码实现是，给 ScoreDaoImpl 类增加一个名为 getAvgScoreById 的方法输入学生学号，返回该生的平均成绩，该方法对数据库的访问用前面创建的带输入输出的存储过程 proc proc_get_avgscore_id 实现，该方法实现代码如下：

```
public int getAvgScoreById(String sid) {
            Connection conn = null;
            CallableStatement cstmt = null;
            ResultSet rs = null;
            conn = BaseDao.getConn();
            int avg = 0;
            try {cstmt = conn.prepareCall("{call dbo.proc_get_avgscore_id
(?,?)}");
                  cstmt.setString(1, sid);
                  cstmt.registerOutParameter(2, java.sql.Types.INTEGER);
                  cstmt.execute();
                  avg = cstmt.getInt(2);
            } catch (SQLException e) {
                  e.printStackTrace();
            } finally {
                  try {
                        if (rs != null)
                              rs.close();
                        if (cstmt != null)
                              cstmt.close();
                        if (conn != null)
                              conn.close();
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
            return avg;
      }
```

说明：输出参数的类型用 java.sql.Types 来指定。

【例 4】创建带游标的存储过程 proc_select_student_includeCursor，实现查询所有不及格门数超过 4 门的学生的学号、姓名、不及格的课程名和成绩插入到 notpasscout 表中。先

建立一张名为 notpasscout 的表，具体 SQL 指令如下：

```
use  ssms
go
create table notpasscout
    ( sid char(10),
     sname varchar(10),
    cname varchar(60),
    totalscore int
    )
```

创建存储过程指令如下：

```
use  ssms
go
if(object_id('proc_select_student_includeCursor', 'p') is not null)
drop proc proc_select_student_includeCursor
go
create proc proc_select_student_includeCursor
as
begin
delete from notpasscout—先删除 notpasscout 表中已有的数据
DECLARE notPassCursor CURSOR LOCAL SCROLL FOR -声明游标
    select sid,count(*) 'nopasscount'
    from score
    where totalscore <60
    group by sid
OPEN notPassCursor
declare @sid char(10),@nopasscount int
FETCH NEXT FROM notPassCursor INTO @sid,@nopasscount—读取游标中数据到变量中
WHILE @@FETCH_STATUS = 0
BEGIN
    if(@nopasscount>=4)
    begin
    insert into notpasscout—把从数据库中读取的数据插入到 notpasscout 中
    select score.sid,sName,cName,totalScore
    from student,score,course
    where student.sId=score.sId
    and course.cId=score.cId
    and student.sId=@sid
    and totalScore<60
    end
FETCH NEXT FROM notPassCursor INTO @sid,@nopasscount—读取游标中下一条数据
END
CLOSE notPassCursor
end
```

上面创建的存储过程“proc_select_student_includeCursor”是带游标无参的存储过程，所以 JSP 调用该存储过程方法同 JSP 调用无参存储过程，只不过该存储过程没有返回值，PrepareStatement 需要执行 execute()方法，来执行存储过程，而不是执行 executeQuery()方

法。具体实现代码如下：

```
   public void createNotPass (){
              Connection conn = null;
              PreparedStatement stmt = null;
              ResultSet rs=null;
              try
{Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
              conn = DriverManager.getConnection(
              "jdbc:sqlserver://localhost:1433;DatabaseName=ssms",      "sa",
"hh123456");
   stmt      =      conn.prepareStatement("{call      dbo.proc_select_student_
includeCursor}");
                     stmt.execute();
              } catch (SQLException e) {
                     e.printStackTrace();
              } catch (ClassNotFoundException e) {
                     e.printStackTrace();
              } finally {
                     try {
                            if (rs != null)
                                   rs.close();
                            if (stmt != null)
                                   stmt.close();
                            if (conn != null)
                                   conn.close();
                     } catch (Exception e) {
                            e.printStackTrace();
                     }
              }
              }
```

应用 5　同时具有返回值、输入参数、输出参数的存储过程

【例】创建存储过程 pro_get_student3 实现随机输入学生学号查询该生是否成年及其个人姓名、性别信息。

```
   if(OBJECT_ID('proc_get_Student3','p') is not null)
   drop proc proc_get_Student3
   go
   create proc proc_get_Student3
   (@sno char(10),
   @name varchar(10) output,
   @sex char(2) output
   )
   as
   declare @c int
   select @name=sName,@sex=sSex from view_age where sid=@sno and Sage>=18
   set @c=@@ROWCOUNT
   return @c
```

```
go
declare @name varchar(10),@sex char(2),@count int
exec @count=proc_get_Student3 '1334110101' ,@name output,@sex output
select @name,@sex
select @count
```

应用 6　同时返回参数和记录集的存储过程

【例】创建存储过程 pro_get_student4 实现按性别查询成年学生信息，同时返回人数。

```
if(OBJECT_ID('proc_get_Student4','p') is not null)
drop proc proc_get_Student4
go
create proc proc_get_Student4
(
@sex char(2)
)
as
declare @c int
select * from view_age where sSex=@sex
set @c=@@ROWCOUNT
return @c
go
declare @count int
exec @count=proc_get_Student4 '男'
select @count
```

应用 7　返回多个记录集的存储过程

【例】创建存储过程 pro_get_info 实现查询所有学生成绩信息及及格分以上学生成绩信息。

```
if(object_id('pro_get_info', 'p') is not null)
drop proc pro_get_info
go
create proc pro_get_info
(
@lowscore int,
@highscore int
)
as
select student.sid,sName,course.cid,cName,totalScore
from student,course,score where student.sid=score.sid and course.cid=
score.cid
select student.sid,sName,course.cid,cName,totalScore
from student,course,score
where student.sid=score.sid and course.cid=score.cid and totalScore between
@lowscore and @highscore
order by totalScore
go
```

```
exec pro_get_info 60,100
```

任务 5.1　建立成绩业务处理模型

一、建立 ScoreImpl 接口

按照前面建立接口的方法，在 com.ssms.biz 包中建立 ScoreImpl 接口，其代码如下：

```
package com.ssms.biz;
import java.util.List;
import com.ssms.entity.NullScore;
import com.ssms.entity.Score;
public interface ScoreBiz {
    public List<Score> getAllScore();
    public Score getOneScoreById( String sno,String cno,String tid);
    public List<Score> getScoreByTid( String tid);
    public List<Score> getScoreBySidAndTid( String sid,String tid);
    public List<Score> getScoreBySnameAndTid(String sname,String tid);
    public List<Score> getScoreByTotalScorePrecise(int totalscore,String
tid);
    public List<Score> getScoreByTotalScoreCompare(String totalscore,
String tid);
    public List<Score> getScoreByTotalScoreSet(String totalscore,String
tid);
    public List<Score> getScoreByTotalScoreScope(String tid,int begin,int
end);
    public List<NullScore> getScoreByTotalScoreNull(String tid);
    public int updateScore(String sno, String cname, int usualscore,int
experimentscore,int examscore,int totalscore);
    public boolean isDeleteScore(String sid, String cid, String tid);
    public boolean isAddScore(String sid,String cid,String tid,int
usualScore,int experimentScore,int examScore,int totalScore);
    public boolean isBackupScore(String path);
    public boolean isRestoreScore(String path);
    }
```

二、建立 ScoreBizImpl 类

按照前面建立类的方法，在 com.ssms.biz.impl 包中建立 ScoreBizImpl 类，实现 ScoreBiz 接口，其代码如下：

```
package com.ssms.biz.impl;
import java.util.List;
import com.ssms.biz.ScoreBiz;
import com.ssms.dao.ScoreDao;
import com.ssms.dao.impl.ScoreDaoImpl;
import com.ssms.entity.NullScore;
import com.ssms.entity.Score;
public class ScoreBizImpl implements ScoreBiz {
```

```
       public List<Score> getAllScore() {
       ScoreDao scdao=new ScoreDaoImpl();
       return scdao.getAllScore();
       }
       public Score getOneScoreById(String sno, String cno, String tid) {
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getOneScoreById(sno, cno, tid);
       }
              public List<Score> getScoreByTid(String tid) {
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getScoreByTid(tid);
       }
       public List<Score> getScoreBySidAndTid(String sid, String tid) {
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getScoreBySidAndTid(sid, tid);
       }
       public List<Score> getScoreBySnameAndTid(String sname, String tid) {
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getScoreBySnameAndTid(sname, tid);
       }
       public List<Score> getScoreByTotalScorePrecise(int totalscore, String
tid) {
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getScoreByTotalScorePrecise(totalscore, tid);
       }
       public  List<Score>  getScoreByTotalScoreCompare(String  totalscore,
String tid) {
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getScoreByTotalScoreCompare(totalscore, tid);
       }
      public List<Score> getScoreByTotalScoreSet(String totalscore, String tid)
{
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getScoreByTotalScoreSet(totalscore, tid);
       }
       public List<Score> getScoreByTotalScoreScope(String tid, int begin, int
end) {
              ScoreDao scdao=new ScoreDaoImpl();
           return scdao.getScoreByTotalScoreScope(tid, begin, end);
       }
       public List<NullScore> getScoreByTotalScoreNull(String tid) {
              ScoreDao scdao=new ScoreDaoImpl();
              return scdao.getScoreByTotalScoreNull(tid);
       }
       public int updateScore(String sno, String cname, int usualscore, int
experimentscore, int examscore,         int totalscore) {
              ScoreDao scdao=new ScoreDaoImpl();
             return scdao.updateScore(sno, cname, usualscore, experimentscore,
examscore, totalscore);
       }
```

```
        public boolean isDeleteScore(String sid, String cid, String tid) {
                ScoreDao scdao=new ScoreDaoImpl();
                return scdao.isDeleteScore(sid, cid, tid);
        }
        public boolean isAddScore(String sid, String cid, String tid, int
usualScore, int experimentScore, int examScore,             int totalScore) {
                ScoreDao scdao=new ScoreDaoImpl();
    return  scdao.isAddScore(sid,  cid,  tid,  usualScore,  experimentScore,
examScore, totalScore);
        }
        public boolean isBackupScore(String path) {
                ScoreDao scdao=new ScoreDaoImpl();
                return scdao.isBackupScore(path);
        }
        public boolean isRestoreScore(String path) {
                ScoreDao scdao=new ScoreDaoImpl();
                return scdao.isRestoreScore(path);
        }
    }
```

任务 5.2　建立教师数据处理控制器 TeacherSearchScore

在 com.ssms.server 包中建立名为 TeacherSearchScore 的 servlet，其具体代码如下：

```
    package com.ssms.server;
    import java.io.IOException;
    import javax.servlet.ServletException;
    import javax.servlet.annotation.WebServlet;
    import javax.servlet.http.HttpServlet;
    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;
    import javax.servlet.http.HttpSession;
    import java.util.ArrayList;
    import java.util.List;
    import com.ssms.dao.ScoreDao;
    import com.ssms.dao.impl.ScoreDaoImpl;
    import com.ssms.entity.NullScore;
    import com.ssms.entity.Score;
    import com.ssms.entity.Teacher;
    @WebServlet("/TeacherSearchScore")
    public class TeacherSearchScore extends HttpServlet {
        private static final long serialVersionUID = 1L;
        public TeacherSearchScore() {
                super();
                        }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
                doPost(request, response);
        }
    protected  void  doPost(HttpServletRequest  request,  HttpServletResponse
```

```
response)
                    throws ServletException, IOException
   {          request.setCharacterEncoding("utf-8");
              response.setCharacterEncoding("utf-8");
              HttpSession session = request.getSession(true);
              ScoreDao score = new ScoreDaoImpl();
              Teacher t = new Teacher();
              t=(Teacher)session.getAttribute("user");
              List<Score> scorelist = new ArrayList <Score>();
              List<NullScore> nullscorelist = new ArrayList <NullScore>();
              scorelist=   score.getScoreByTid( t.getTid());
              String selectitem = "", selectscoreitem = "";
              if (request.getParameter("select") != null)
                     selectitem  =  request.getParameter("select").toString().
trim();
              if (request.getParameter("selectscoretype") != null)
   selectscoreitem   =   request.getParameter("selectscoretype").toString().
trim();
              String selectcontent ="";
              if (request.getParameter("searchcontent") != null)
   selectcontent = request.getParameter("searchcontent").toString().trim();
              if (request.getParameter("select") != null)
              {    if (selectitem.equals("学号") || selectitem.equals("姓名"))
                     {      if (!("".equals(selectcontent) || selectcontent ==
null))
                            {             if (selectitem.equals("学号"))
       {      scorelist       =       score.getScoreBySidAndTid(selectcontent,
t.getTid());
                                                  }
                                   else if (selectitem.equals("姓名"))
   {scorelist = score.getScoreBySnameAndTid(selectcontent, t.getTid());
                                   }
                            }
                     }
                                    else if (selectitem.equals("综合成绩"))
                                          {      String searchscorecontent ="";
                                                 int searchscorecontentint=0;
                                          int beginscore = 0, endscore = 0;
   if (request.getParameter("searchscorecontent") != null)

       searchscorecontent=request.getParameter("searchscorecontent").toString().
trim();
              if(!("".equals(searchscorecontent) || searchscorecontent == null))
              {      if (selectscoreitem.equals("精确查找"))

       {  searchscorecontentint=Integer.valueOf(searchscorecontent);
   scorelist   =   score.getScoreByTotalScorePrecise(searchscorecontentint,t.
getTid());
       }else if (selectscoreitem.equals("比较查找"))
   scorelist    =    score.getScoreByTotalScoreCompare(searchscorecontent,t.
```

```
getTid());
    else if (selectscoreitem.equals("集合查找"))
    scorelist = score.getScoreByTotalScoreSet(searchscorecontent,t.getTid());
        }
        else if (selectscoreitem.equals("无成绩查找"))
        {nullscorelist = score.getScoreByTotalScoreNull(t.getTid());

        session.setAttribute("nullscorelist",nullscorelist);}
                                                    else   if   (selectscoreitem.
Equals ("范围查找"))
                              {     if(request.getParameter("beginscore")!=null
&& !("".equals(request.getParameter("beginscore").toString().trim())))
    beginscore = Integer.valueOf((request.getParameter("beginscore").toString().
trim()));
        if(request.getParameter("endscore")!=null&&!("".equals(request.getPar
ameter("endscore").toString().trim())))
    endscore
=Integer.valueOf(request.getParameter("endscore").toString().trim());
                              if(beginscore!=0 && endscore!=0)
        scorelist   =   score.getScoreByTotalScoreScope(t.getTid(),beginscore,
endscore);
                                                              }

                                                  }

                      }
                session.setAttribute("searchscore", scorelist);
    request.getRequestDispatcher("teachersearchscore.jsp").forward(request,
response);
        }
    }
```

任务 5.3　建立 teachersearchscore.jsp 页面

在 WebContent 文件夹下建立 teachersearchscore.jsp 页面，其代码如下：

```
    <%@page import="com.ssms.entity.NullScore"%>
    <%@page import="com.ssms.entity.Score"%>
    <%@page import="com.ssms.dao.ScoreDao"%>
    <%@page import="com.ssms.dao.impl.ScoreDaoImpl"%>
    <%@page import="java.util.*"%>
    <%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>查询所授课程的成绩信息</title>
    </head>
```

```
   <body>
      <script language="javascript" charset="UTF-8">
            function change() {
                  var option = document.getElementById("select").options;
   var scoreoption = document.getElementById("selectscoretype").options;
                  for (var i = 0; i < option.length; i++)
                        option[i].value = option[i].innerHTML;
                  for (var j = 0; j < scoreoption.length; j++)
                        scoreoption[j].value = scoreoption[j].innerHTML;
                  document.teacher.submit();
            }
            function changeItem() {
                  var selectitem = teacher.select.value;
                  window.location.href = "teachersearchscore.jsp?select="
                              + selectitem;
            }
            function scorechangeItem() {
                  var selectitem = teacher.select.value;
                  var selectscoreitem = teacher.selectscoretype.value;
                  window.location.href = "teachersearchscore.jsp?select="
            + selectitem + "&selectscoretype=" + selectscoreitem;
            }
      </script>
      <%     request.setCharacterEncoding("UTF-8");
            response.setCharacterEncoding("UTF-8");
            String selectitem = "学号", selectscoreitem = "精确查找";
            if (request.getParameter("select") != null)
                  selectitem  =  request.getParameter("select").toString().
trim();
            if (request.getParameter("selectscoretype") != null)
                 selectscoreitem = request.getParameter("selectscoretype").
toString().trim();
            ScoreDao score = new ScoreDaoImpl();//业务类
            List<Score> searchscore = new ArrayList<Score>();
            searchscore = (List<Score>) session.getAttribute("searchscore");
            Score currentScore = new Score();//生成一个实体对象
      %>
      <form action="TeacherSearchScore" name="teacher" method="post">
            请选择查询条件：        <select
                  style='font-size:    18px'    name="select"    onChange=
"changeItem()">
   <option value="学号" <%if (selectitem.equals("学号")) {%> selected <%}%>>
学号</option>
   <option value="姓名" <%if (selectitem.equals("姓名")) {%> selected <%}%>>
姓名</option>
   <option value="综合成绩" <%if (selectitem.equals("综合成绩")) {%> selected
                        <%}%>>综合成绩</option>
            </select>      
            <%    if (selectitem.equals("学号") || selectitem.equals("姓名"))
{
```

```
            %>
      <input type="text" name="searchcontent" style="width: 160px" value="" />
                  
            <%          } else if (selectitem.equals("综合成绩")) {
   %>请选择成绩查找方式:     <select style='font-size: 18px'
                  name="selectscoretype" onChange="scorechangeItem()">
   <option value="精确查找" <%if (selectscoreitem.equals("精确查找")) {%>
                  selected <%}%>>精确查找</option>
      <option value="比较查找" <%if (selectscoreitem.equals("比较查找")) {%>
                  selected <%}%>>比较查找</option>
      <option value="范围查找" <%if (selectscoreitem.equals("范围查找")) {%>
                  selected <%}%>>范围查找</option>
      <option value="集合查找" <%if (selectscoreitem.equals("集合查找")) {%>
                  selected <%}%>>集合查找</option>
      <option value="无成绩查找" <%if (selectscoreitem.equals("无成绩查找")) {%>
                  selected <%}%>>无成绩查找</option>
            </select>      
            <%     }
                  if ((selectitem.equals("综合成绩")) && ((selectscoreitem.
equals("精确查找")) || (selectscoreitem.equals("比较查找"))
                        || (selectscoreitem.equals("集合查找")))) {
            %>
            请输入查找的成绩条件:  <input type="text" name="searchscorecontent"
width=200
                  style="width: 40px" />
            <%     }
   if ((selectitem.equals("综合成绩")) && (selectscoreitem.equals("范围查找")))
{
            %>
            查找成绩开始值<input type="text" name="beginscore" width=200
   style="width: 40px" />     成绩终止值<input
                  type="text" name="endscore" width=200 style="width: 40px"
/>
            <%     }
            %>
            <input type="submit" value="查找"
                  style='font-size: 20px; width: 80px; height: 30px;' /><br>
<br>   <br>
            <table border="2" width="900">
                  <tr>   <td width="80">学号</td>
                        <td width="100">姓名</td>
                        <td width="160">课程名</td>
                        <td width="80">班级</td>
                        <td width="80">平时成绩</td>
                        <td width="80">实验成绩</td>
                        <td width="80">考试成绩</td>
                        <td width="80">综合成绩</td>
                        <td width="80">操作</td>
                        <td width="80">操作</td>
                  </tr>
```

```
                    <%if (selectscoreitem.equals("无成绩查找"))
      {List<NullScore> searchnullscore = new ArrayList<NullScore>();
                    NullScore currentnullscore=new NullScore();
      searchnullscore     =     (List<NullScore>)     session.getAttribute
("nullscorelist");
                    if(searchnullscore!=null){
                    for (int i = 0; i < searchnullscore.size(); i++) {
                          currentnullscore  =  (NullScore)  searchnullscore.
get(i);
                          %>
                          <tr>
                          <td><%=currentnullscore.getSid()%></td>
                          <td><%=currentnullscore.getSname()%></td>
                          <td><%=currentnullscore.getCname()%></td>
                          <td><%=currentnullscore.getsClassName()%></td>
                          <td><%=currentnullscore.getUsualResults()%></td>

      <td><%=currentnullscore.getExperimentalResults()%></td>
                          <td><%=currentnullscore.getExamResults()%></td>
                          <td><%=currentnullscore.getTotalScore()%></td>
            <td><input  type="button"  value="修改"  style='font-size:  16px'

      onclick="window.location.href('update.jsp?sid=<%=currentScore.getSid
()%>&cname=<%=currentScore.getCname()%>&usualscore=<%=currentScore.getUsual
Results()%>&experimentscore=<%=currentScore.getExperimentalResults()%>&
examscore=<%=currentScore.getExamResults()%>&totalscore=<%=currentScore.
getTotalScore()%>')" /></td>                          <td><input
type="button" value="删除" style='font-size: 16px'
      onclick="window.location.href('delete.jsp?sid=<%=currentScore.getSid()
%>&cid=<%=currentScore.getCid()%>&tid=<%=currentScore.getTid()%>')" /></td>
                    </tr>
                    <%     }
                    %>
   <tr>             <td colspan="5" align="center"><input type="button" value=
"添加"
   style='font-size: 16px' onclick="window.location.href('add.jsp')" /></td>
   <td colspan="5" align="center"><input id="close" type="button"
   style='font-size: 16px' value="关闭" name="close"
            onclick="window.close()"></td>
                    </tr>
                    <%}}
                    else
            {for (int i = 0; i < searchscore.size(); i++) {
      currentScore = (Score) searchscore.get(i);//取出集合中的每个对象
                    %>
                    <tr>
                          <td><%=currentScore.getSid()%></td>
                          <td><%=currentScore.getSname()%></td>
                          <td><%=currentScore.getCname()%></td>
                          <td><%=currentScore.getsClassName()%></td>
```

```
                        <td><%=currentScore.getUsualResults()%></td>
                    <td><%=currentScore.getExperimentalResults()%></td>
                        <td><%=currentScore.getExamResults()%></td>
                        <td><%=currentScore.getTotalScore()%></td>
            <td><input type="button" value="修改" style='font-size: 16px'

      onclick="window.location.href('update.jsp?sid=<%=currentScore.getSid()
%>&cname=<%=currentScore.getCname()%>&usualscore=<%=currentScore.getUsual
Results()%>&experimentscore=<%=currentScore.getExperimentalResults()%>&
examscore=<%=currentScore.getExamResults()%>&totalscore=<%=currentScore.
getTotalScore()%>')" /></td>
      <td><input type="button" value="删除" style='font-size: 16px'

      onclick="window.location.href('delete.jsp?sid=<%=currentScore.getSid()
%>&cid=<%=currentScore.getCid()%>&tid=<%=currentScore.getTid()%>')" /></td>
                    </tr>
                    <%     }      %>
                    <tr>
      <td colspan="5" align="center"><input type="button" value="添加"
   style='font-size: 16px' onclick="window.location.href('add.jsp')" /></td>
   <td colspan="5" align="center"><input id="close" type="button"
   style='font-size: 16px' value="关闭" name="close"
            onclick="window.close()"></td>
                    </tr>
                    <%     } %>
            </table>
      </form>
   </body>
   </html>
```

任务 5.4　修改成绩

1. 建立 update.jsp 页面

在 webContent 文件夹建立 update.jsp 页面，其具体代码如下：

```
   <%@page import="com.ssms.entity.Teacher"%>
   <%@page import="java.sql.ResultSet"%>
   <%@page import="java.sql.Statement"%>
   <%@page import="java.sql.Connection"%>
   <%@page import="java.sql.DriverManager"%>
   <%@page import="java.sql.PreparedStatement"%>
   <%@page import="java.sql.SQLException"%>
   <%@ page language="java" contentType="text/html; charset=utf-8"
      pageEncoding="utf-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
   <title>修改学生成绩</title>
   </head>
   <body>
      <form name="update" method="post" action="TeacherUpdateScore">
            <%     request.setCharacterEncoding("UTF-8");
                   response.setCharacterEncoding("utf-8");
                   String sid = request.getParameter("sid");
                   String cname = request.getParameter("cname");
   int usualscore=Integer.valueOf(request.getParameter("usualscore").toString().
trim());
   int  experimentscore=Integer.valueOf(request.getParameter("experimentscore").
toString().trim());
   int  examscore=Integer.valueOf(request.getParameter("examscore").toString().
trim());
   int totalscore=Integer.valueOf(request.getParameter("totalscore").toString().
trim());
                   %>
            <table align="center" width="600">
                   <tr>   <td colspan="4" align="center">
                                 <h2>请输入要修改的学生成绩信息</h2>
                                 <hr>
                          </td>
                   </tr>
                   <tr>   <td>学 号：</td>
                          <td><input type="text" name="updateSid" width=200
                                 readOnly  value=<%=sid%> /></td>
                          <td>课程名：</td>
            <td><input type="text" name="updateCname" width=200
                                 readOnly  value=<%=cname%> /></td>
                   </tr>  <tr>   </tr>  <tr>   </tr>
                   <tr>          <td>平时成绩：</td>
            <td><input type="text" name="updateUsualResults" width=200
                                 value=<%=usualscore%> /></td>
                          <td>实验成绩：</td>
            <td><input type="text" name="updateExperimentalResults"
                   width=200 value=<%=experimentscore%> /></td>
                   </tr>  <tr>   </tr><tr>    </tr>
                   <tr>   <td>考试成绩：</td>
            <td><input type="text" name="updateExamResults" width=200
                                 value=<%=examscore%> /></td>
                          <td>综合成绩：</td>
            <td><input type="text" name="updateTotalScore" width=200
                                 value=<%=totalscore%> /></td>
                   </tr>  <tr>   </tr><tr>    </tr>
      <tr>   <td colspan="2" align="center"><input type="submit" value="修改"
                                 style='font-size: 18px' /></td>
      <td colspan="2" align="center"><input id="close" type="button"
      style='font-size: 18px' value="关闭" name="close"
                                 onclick="window.close()"></td>
                   </tr>
```

```
            </table>
        </form>
    </body>
    </html>
```

2. 建立 TeacherUpdateScore 控制器

在 com.ssms.server 文件夹下建立名为 TeacherUpdateScore 的 servlet，其具体代码如下：

```
    package com.ssms.server;
    import java.io.IOException;
    import java.io.PrintWriter;
    import javax.servlet.ServletException;
    import javax.servlet.annotation.WebServlet;
    import javax.servlet.http.HttpServlet;
    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;
    import com.ssms.dao.ScoreDao;
    import com.ssms.dao.impl.ScoreDaoImpl;
    @WebServlet("/TeacherUpdateScore")
    public class TeacherUpdateScore extends HttpServlet {
        private static final long serialVersionUID = 1L;
        public TeacherUpdateScore() {
            super();
            }
        protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            doPost(request,response);
        }
        protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            request.setCharacterEncoding("UTF-8");
            response.setContentType("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            String sid = request.getParameter("updateSid");
            String cname = request.getParameter("updateCname");
            int    usualscore    =    Integer.valueOf(request.getParameter
("updateUsualResults").toString().trim());
            int   experimentscore   =   Integer.valueOf(request.getParameter
("updateExperimentalResults").toString().trim());
            int    examscore    =    Integer.valueOf(request.getParameter
("updateExamResults").toString().trim());
            int    totalscore    =    Integer.valueOf(request.getParameter
("updateTotalScore").toString().trim());
            ScoreDao  score= new ScoreDaoImpl();
           int id = score.updateScore(sid, cname, usualscore, experimentscore,
examscore, totalscore);
            if (id!=-1) {
                  out.println("<script>alert('成功修改成绩！');window.
location.href ='TeacherSearchScore';</script>");
```

```
            } else {            out.println("<script>alert('成绩修改失败，请重
新修改！');window.location.href ='update.jsp';</script>");
                }
        }
    }
```

任务 5.5　删除成绩

1. 建立 delete.jsp 页面

```
    <%@page import="com.ssms.entity.Score"%>
    <%@page import="com.ssms.dao.impl.ScoreDaoImpl"%>
    <%@page import="com.ssms.dao.ScoreDao"%>
    <%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>删除成绩</title>
    <script type="text/javascript">
        function del() {
              var flag = false;
              if (confirm("您确定要删除吗？")) {
                    flag = true; }
              return flag;
        }
    </script>
    </head>
    <body>
        <%      request.setCharacterEncoding("UTF-8");
              response.setCharacterEncoding("utf-8");
              String sid = request.getParameter("sid").toString().trim();
              String cid = request.getParameter("cid").toString().trim();
              String tid = request.getParameter("tid").toString().trim();
              ScoreDao dao = new ScoreDaoImpl();
              Score sc = dao.getOneScoreById(sid, cid, tid);
        %>
        <H2 align="center">删除的成绩信息</H2>
        <hr>
        <form action="DeleteScore" method="post" onsubmit="return del()">
              <table border="1" class="font" align="center">
                    <tr align="center">
                          <td>学号：</td>
              <td><input type="text" readonly="readonly" name="sid"
                                value="<%=sid%>"></td>
                    </tr>
                    <tr align="center">
                          <td>学生姓名：</td>
```

```
                    <td><input type="text" readonly="readonly" name="sname"
                                value="<%=sc.getSname()%>"></td>
                    </tr>
                    <tr align="center">
                          <td>课程号：</td>
                    <td><input type="text" readonly="readonly" name="cid"
                                value="<%=sc.getCid()%>"></td>
                    </tr>
                    <tr align="center">
                          <td>课程名：</td>
             <td><input type="text" readonly="readonly" name="cname"
                                value="<%=sc.getCname()%>"></td>
                    </tr>
                    <tr align="center">
                          <td>教师编号：</td>
             <td><input type="text" readonly="readonly" name="tid"
                                value="<%=tid%>"></td>
                    </tr>
                    <tr align="center">
                          <td>平时成绩:</td>
             <td><input type="text" readonly="readonly" name="usualresult"
                    value="<%=sc.getUsualResults()%>"></td>
                    </tr>
                    <tr align="center">
                          <td>实验成绩:</td>
             <td><input type="text" readonly="readonly"
name="experimentresult" value="<%=sc.getExperimentalResults()%>">
                          </td>
                    </tr>
                    <tr align="center">
                          <td>考试成绩:</td>
             <td><input type="text" readonly="readonly" name="examresult"
                          value="<%=sc.getExamResults()%>"></td>
                    </tr>
                    <tr align="center">
                          <td>综合成绩:</td>
             <td><input type="text" readonly="readonly" name="totalresult"
                                value="<%=sc.getTotalScore()%>"></td>
                    </tr>
                    <tr align="center">
             <td><input type="submit" value="删除">  </td>
                          <td><input type="button" value="取消"

    onclick="window.location.href='teachersearchscore.jsp'"></td>
                    </tr>
              </table>
    </form>
</body>
</html>
```

2. 建立 DeleteScore 删除成绩控制器

按照前面建立servlet的位置、方法和步骤建立名为DeleteScore的servlet，其内容如下：

```
package com.ssms.server;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.ssms.biz.ScoreBiz;
import com.ssms.biz.impl.ScoreBizImpl;
@WebServlet("/DeleteScore")
public class DeleteScore extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public DeleteScore() {
            super();
            // TODO Auto-generated constructor stub
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
                    throws ServletException, IOException {
            doPost(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
            request.setCharacterEncoding("UTF-8");
            response.setCharacterEncoding("UTF-8");
            String sid = request.getParameter("sid").toString().trim();
            String cid = request.getParameter("cid").toString().trim();
            String tid = request.getParameter("tid").toString().trim();
            ScoreBiz biz = new ScoreBizImpl();
            boolean ret = biz.isDeleteScore(sid, cid, tid);
            if (ret) {    response.sendRedirect("TeacherSearchScore");
            } else {
                    response.sendRedirect("TeacherSearchScore");
            }
    }
}
```

任务 5.6　添加成绩

1. 建立 add.jsp 页面

在 WebConTent 文件夹下，建立名为 add.jsp 页面，其具体代码如下：

```
<%@page import="com.ssms.entity.Teacher"%>
```

```
    <%@ page language="java" contentType="text/html; charset=utf-8"
       pageEncoding="utf-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>增加学生成绩</title>
    </head>
    <body>
    <% String tid=request.getParameter("tid");
    Teacher t=new Teacher();
    t=(Teacher)session.getAttribute("user");%>
       <form name="addform" method="post" action="TeacherAddScore">
              <table align="center" width="360">
                     <tr>   <td colspan="4" align="center">
                                   <h3>请输入要添加的学生成绩信息</h3>
                                   <hr color="red">
                            </td>
                     </tr>
                     <tr>   <td>学 号: </td>
              <td><input type="text" name="addSid" width=200 /></td>
                     </tr>  <tr>   </tr>  <tr></tr>
                     <tr>          <td>课程号: </td>
       <td><input type="text" name="addCid" width=200 /></td>
                     </tr>  <tr>   </tr><tr></tr>
                     <tr>   <td>教师编号: </td>
       <td><input type="text" name="addTid" width=200  value=<%=t.getTid()%>
readonly/></td>
                     </tr>  <tr>   </tr>  <tr></tr>
                     <tr>          <td>平时成绩: </td>
    <td><input type="text" name="addUsualResults" width=200 /></td>
                     </tr>  <tr>   </tr>  <tr></tr>
                     <tr>          <td>实验成绩: </td>
    <td><input type="text" name="addExperimentalResults" width=200 /></td>
                     </tr>  <tr>   </tr>  <tr></tr>
                     <tr>   <td>考试成绩: </td>
    <td><input type="text" name="addExamResults" width=200 /></td>
                     </tr><tr>     </tr><tr></tr>
                     <tr>   <td>综合成绩: </td>
    <td><input type="text" name="addTotalScore" width=200 /></td>
                     </tr><tr>     </tr>  <tr></tr>
    <tr><td colspan="2" align="center"><input type="submit" value="添加"
                                  style='font-size: 18px' />   
      
              <input id="close" type="button"
    style='font-size: 18px' value="关闭" name="close"
                                   onclick="window.close()"></td>
                     </tr>
              </table>
```

```
      </form>
   </body>
   </html>
```

2. 建立 TeacherAddScore 成绩添加控制器

按照前面建立 servlet 的位置、方法和步骤建立名为 TeacherAddScore 的 servlet，其代码如下：

```
   package com.ssms.server;
   import java.io.IOException;
   import java.io.PrintWriter;
   import javax.servlet.ServletException;
   import javax.servlet.annotation.WebServlet;
   import javax.servlet.http.HttpServlet;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;
   import com.ssms.biz.ScoreBiz;
   import com.ssms.biz.impl.ScoreBizImpl;
   @WebServlet("/TeacherAddScore")
   public class TeacherAddScore extends HttpServlet {
      private static final long serialVersionUID = 1L;
        public TeacherAddScore() {
          super();
           }
      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            response.getWriter().append("Served   at:   ").append(request.
getContextPath());
      }
      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
          request.setCharacterEncoding("UTF-8");
          response.setCharacterEncoding("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
          String sid=request.getParameter("addSid");
          String cid=request.getParameter("addCid");
          String tid=request.getParameter("addTid");
   int  usualScore=Integer.valueOf(request.getParameter("addUsualResults").
toString().trim());
   int experimentScore=Integer.valueOf(request.getParameter ("addExperimentalResults").
toString().trim());
   int    examScore=Integer.valueOf(request.getParameter("addExamResults").
toString().trim());
   int    totalScore=Integer.valueOf(request.getParameter("addTotalScore").
toString().trim());
          ScoreBiz biz=new ScoreBizImpl();
          Boolean ret=false;
```

```
   ret=biz.isAddScore(sid, cid, tid, usualScore, experimentScore, examScore,
totalScore);
          if (ret) {
   out.println("<script>alert(' 成 功 添 加 成 绩 ！ ');window.location.href  =
'TeacherSearchScore';</script>");
             }else{    out.println("<script>alert('添加成绩失败，请重新添加！');
window.location.href ='add.jsp';</script>");
                    }
       }
   }
```

任务 5.7　修改教师密码

1. 修改 TeacherDao 接口

添加下面一行代码：

```
      public boolean isUpdatePass(String tid,String pwd);
```

2. 修改 TeacherDaoImpl

添加如下方法：

```
   public boolean isUpdatePass(String tid, String newpass) {
             boolean ret=false;
             Connection conn=null;
             PreparedStatement pstmt=null;
           conn=BaseDao.getConn();
   newpass=MD5Encode.getMD5(newpass);
            String sql="update teacher set tpassword=? where tid=?";
       try {
             pstmt=conn.prepareStatement(sql);
             pstmt.setString(1,tid);
             pstmt.setString(2,newpass);
             int num=pstmt.executeUpdate();
              if(num!=0){ret=true;}
      } catch (SQLException e) {
             e.printStackTrace();
      } finally {
             BaseDao.closeAll(null, pstmt, conn);
      }
             return ret;
      }
```

3. 修改 TeacherBiz 接口

在 TeacherBiz 接口中添加如下一行代码：

```
   public boolean isUpdatePass(String tid, String newpass);
```

4. 修改 TeacherBizImpl 类

在 TeacherBizImpl 类中添加如下方法：

```
   public boolean isUpdatePass(String tid, String newpass) {
                  return tdao.isUpdatePass(tid, newpass);
            }
```

5. 建立 updatepass.jsp 页面

按照前面介绍的建立 jsp 的位置、方法和步骤建立名为 updatepass.jsp 的页面，其代码如下：

```
   <%@page import="com.ssms.entity.Admin"%>
   <%@page import="com.ssms.entity.Teacher"%>
   <%@page import="java.sql.SQLException"%>
   <%@page import="java.sql.ResultSet"%>
   <%@page import="java.sql.Statement"%>
   <%@page import="java.sql.Connection"%>
   <%@page import="java.sql.DriverManager"%>
   <%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>密码修改</title>
   </head>
   <body>
      <script type="text/javascript">
            function isValid() {
                  if (document.change.oldpass.value.length == 0) {
                        conForm = false;
                        window.alert('必须输入原密码');
                        doucument.change.elements(1).focus();
                        return false;
                  } else if (document.change.oldpass.value != document.
change.pws.value) {                    document.change.oldpass.value = "";
                        window.alert('输入的原始密码不正确 ');
                        doucument.change.elements(1).focus();
                        return false;
                  } else if (document.change.newpass.value.length == 0) {
                        window.alert('必须输入现在的密码');
                        doucument.change.elements(2).focus();
                        return false;
                  } else if (document.change.newpass.value != document.
change.surepass.value) {
                        document.change.surepass.value = "";
                        window.alert('两次输入的密码不一样');
```

```
                        doucument.change.elements(3).focus();
                        return false;
                    }
                    return true;
            }
    </script>
    <%      String uid = request.getParameter("id");
            String uname =request.getParameter("name");
            String pws = request.getParameter("pws");
    %>
    <form name="change" method="post" id="ppp" action="UpdatePass">
            <h3 align="center">修改个人密码</h3>
            <hr color="red">
            <table align="center">
                    <tr>
                        <td>用户名：</td>
<td><input type="text" value="<%=uname%>" width="200" readonly /></td>
                    </tr>
                    <tr>          <td>账号：</td>
    <td><input type="text" name="userid" value="<%=uid%>"
                                width="200" readonly></td>
                    </tr>
                    <tr>          <td></td>
    <td><input type="hidden" name="pws" value="<%=pws%>"
                                width="200" /></td>
                    </tr>
                    <tr>          <td>旧密码：</td>
<td><input type="password" name="oldpass" width="200"></td>
                    </tr>
                    <tr>          <td>新密码：</td>
    <td><input type="password" name="newpass" width="200"></td>
                    </tr>
                    <tr>   <td>确认密码：</td>
    <td><input type="password" name="surepass" width="200"></td>
                    </tr>
                    <tr align="center">
    <td><input type="button" name="ok" value="确定"
onClick="if(isValid()) document.getElementById('ppp').submit();" /></td>
                        <td><input type="reset" value="重置" /></td>
                    </tr>
            </table>
    </form>
</body>
</html>
```

6. 建立修改密码的控制器 UpdatePass

按照前面介绍的建立 servlet 的位置、方法和步骤建立名为 UpdatePass 的 servlet，其代码如下：

```
   package com.ssms.server;
   import java.io.IOException;
   import java.io.PrintWriter;
   import javax.servlet.ServletException;
   import javax.servlet.annotation.WebServlet;
   import javax.servlet.http.HttpServlet;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;
   import javax.servlet.http.HttpSession;
   import com.ssms.biz.AdminBiz;
   import com.ssms.biz.TeacherBiz;
   import com.ssms.biz.impl.AdminBizImpl;
   import com.ssms.biz.impl.TeacherBizImpl;
   @WebServlet("/UpdatePass")
   public class UpdatePass extends HttpServlet {
      private static final long serialVersionUID = 1L;
        public UpdatePass() {
          super();
      }
      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            doPost(request,response);
      }
      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            request.setCharacterEncoding("UTF-8");
            response.setContentType("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            String uid = request.getParameter("userid");
            String pwd = request.getParameter("newpass");
            boolean rst=false;
            HttpSession session = request.getSession(true);
            String logintype=session.getAttribute("logintype").toString();
            if(logintype.equals("teacher"))
            {  TeacherBiz t=new TeacherBizImpl();
                   rst =t.isUpdatePass(uid, pwd);}
            else if(logintype.equals("admin"))
            {  AdminBiz a=new AdminBizImpl();
             rst =a.isUpdatePass(uid, pwd);}
            if (rst) {
   out.println("<script>alert('成功修改密码!');window.location.href ='mainpage.
jsp';</script>");
                  } else {
   out.println("<script>alert('修改密码失败，请重新修改! ');window.location.href
='updatepass.jsp';</script>");
            }
      }
```

任务 6　管理员功能的设计与实现

一、管理员登录

管理员登录与教师登录实现方法相同，由读者自己完成。

二、管理员功能的实现

管理员查看成绩也和教师查看成绩实现方法类似，只不过管理员可以查看所有课程的成绩，教师只能查看自己担任课程的成绩，在查找成绩时调用方法有所不同，所以管理员查找成绩的实现也由读者自己完成。另外注意管理员不能修改、删除和添加成绩，但是管理员可以备份和恢复数据，下面介绍管理员备份和恢复数据的实现。

三、备份数据库

1. 建立 backupdata.jsp 页面

backupdata.jsp 页面其代码如下：

```
   <%@page import="java.util.Date"%>
   <%@page import="java.sql.SQLException"%>
   <%@page import="java.sql.ResultSet"%>
   <%@page import="java.sql.Statement"%>
   <%@page import="java.sql.Connection"%>
   <%@page import="java.sql.DriverManager"%>
   <%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>备份数据库</title>
   </head>
   <body>
      <script language='javascript'>
            function getFullPath(obj) {
                  if (obj) {
            if (window.navigator.userAgent.indexOf("MSIE") >= 1) //ie
                        {     obj.select();
                              return
document.selection.createRange().text;
                        }  else  if  (window.navigator.userAgent.indexOf
("Firefox") >= 1) //firefox
                        {     if (obj.files) {
            return obj.files.item(0).getAsDataURL();
                              }
                              return obj.value;
```

```
                        }
                        return obj.value;
                    }
                }
        </script>
        <form name="backup" action="Backup" method="post">
                <h2 align="center">备份数据库</h2>
                <br> <br>
                <hr color="blue">
                <br> <br>
                     
             
  
                <span style="font-size: 20px;">请选择备份路径：</span>  
<input
                    type="file" style='font-size: 20px; width: 500px; height:
40px;'

        onchange="document.getElementById('filepath').value=getFullPath(this);"
/>
                     <input type="hidden" id="filepath"
                    name="filepath" /> <br>
                <br> <br>
                   &
nbsp;        
                <input type="submit" value="备份数据库" size="600"
                    style='font-size: 20px; width: 600px; height: 60px; color:
red;' />
                 <input id="close" type="button"
                    style='font-size: 20px; height: 60px; width: 80px;' value=
"关闭"
                    name="close" onclick="window.close()">
        </form>
    </body>
    </html>
```

2. 建立数据备份控制器 Backup

数据备份控制器 backup 代码如下：

```
    package com.ssms.server;
    import java.io.IOException;
    import java.io.PrintWriter;
    import java.util.Date;
    import javax.servlet.ServletException;
    import javax.servlet.annotation.WebServlet;
    import javax.servlet.http.HttpServlet;
    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;
    import com.ssms.biz.ScoreBiz;
    import com.ssms.biz.impl.ScoreBizImpl;
```

```
   @WebServlet("/Backup")
   public class Backup extends HttpServlet {
      private static final long serialVersionUID = 1L;
        public Backup() {
          super();
        }
      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            doPost(request,response);
      }
      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            request.setCharacterEncoding("UTF-8");
            response.setCharacterEncoding("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            Date today = new Date();
            @SuppressWarnings("deprecation")
   String date = (today.getYear()) + "年" + (today.getMonth() + 1) + "月" +
(today.getDate()) + "日";
            if (request.getParameter("filepath") != null) {
                  String path = request.getParameter("filepath").toString().
trim()+date + ".bak";
            ScoreBiz biz = new ScoreBizImpl();
            boolean ret = biz.isBackupScore(path);
            if (ret) {   out.println("<script>alert('成功备份数据库！');
window.location.href ='AdminSearchScore';</script>");
                        } else {
                              out.println("<script>alert('备份数据库失败!请
重新备份');window.location.href ='backupdata.jsp';</script>");
                        }
      }
      }
   }
   }
```

四、还原数据库

1. 建立 restoredata.jsp 页面

restoredata.jsp 页面其代码如下：

```
   <%@page import="java.util.Date"%>
   <%@page import="java.sql.SQLException"%>
   <%@page import="java.sql.ResultSet"%>
   <%@page import="java.sql.Statement"%>
   <%@page import="java.sql.Connection"%>
   <%@page import="java.sql.DriverManager"%>
   <%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
```

```
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
   <html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>恢复数据库</title>
   </head>
   <body>
      <script language='javascript'>
            function getFullPath(obj) {
                  if (obj) {
            if (window.navigator.userAgent.indexOf("MSIE") >= 1) //ie
                        {                   obj.select();
            return document.selection.createRange().text;
                        }  else  if  (window.navigator.userAgent.indexOf
("Firefox") >= 1) //firefox
                        {            if (obj.files) {
            return obj.files.item(0).getAsDataURL();
                              }
                              return obj.value;
                        }
                        return obj.value;
                  }
            }
      </script>
      <form name="restore" action="Restore" method="post">
            <h2 align="center">恢复数据库</h2>
            <br> <br>
            <hr color="blue">
            <br> <br>
                 &
nbsp;           
   
            <span style="font-size: 20px;">请选择恢复文件：</span>  
<input
                  type="file" style='font-size: 20px; width: 500px; height:
40px;'

      onchange="document.getElementById('filepath').value=getFullPath(this);"
/>
        <input type="hidden" id="filepath"    name=
"filepath" />
         <br><br> <br>
                 
         
            <input type="submit" value="恢复数据库" size="600"
                  style='font-size: 20px; width: 600px; height: 60px; color:
red;' />
               <input id="close" type="button"
                  style='font-size: 20px; height: 60px; width: 80px;' value=
```

```
"关闭"
                    name="close" onclick="window.close()">
      </form>
   </body>
   </html>
```

2. 建立恢复数据库的控制器 Restore

建立名为 Restore 的 servlet，其代码如下：

```
   package com.ssms.server;
   import java.io.IOException;
   import java.io.PrintWriter;
   import java.util.Date;
   import javax.servlet.ServletException;
   import javax.servlet.annotation.WebServlet;
   import javax.servlet.http.HttpServlet;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;
   import com.ssms.biz.ScoreBiz;
   import com.ssms.biz.impl.ScoreBizImpl;
   @WebServlet("/Restore")
   public class Restore extends HttpServlet {
      private static final long serialVersionUID = 1L;
        public Restore() {
          super();
          // TODO Auto-generated constructor stub
      }
      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            doPost(request,response);
      }
      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            request.setCharacterEncoding("UTF-8");
            response.setCharacterEncoding("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            if (request.getParameter("filepath") != null) {
                  String path = request.getParameter("filepath").toString().
trim();
            ScoreBiz biz = new ScoreBizImpl();
            boolean ret = biz.isRestoreScore(path);
            if (ret) {
                  out.println("<script>alert('成功恢复数据库！');window.
location.href ='AdminSearchScore';</script>");
                  } else {            out.println("<script>alert('恢复数据
库失败! 请重新恢复');window.location.href ='restoredata.jsp';</script>");

                  }
      }
```

```
        }
    }
```

五、管理员修改密码

管理员修改密码同教师修改密码功能的实现，由读者自己完成。

相关知识

一、MVC 设计模式

MVC 是一种使用 MVC（Model View Controller 模型-视图-控制器）设计创建 Web 应用程序的模式，它强制性的使应用程序的输入、处理和输出分开。使用 MVC 应用程序被分成三个核心模块：模型（Model）、视图（View）、控制器（Controller），这三个模块分别充当不同的角色，它们各自处理自己的任务。它们之间彼此又相互联系，构成一个结构分明而又高效的整体。最典型的 MVC 就是 JSP + Servlet + JavaBean 的模式。其结构如图 4-25 所示。

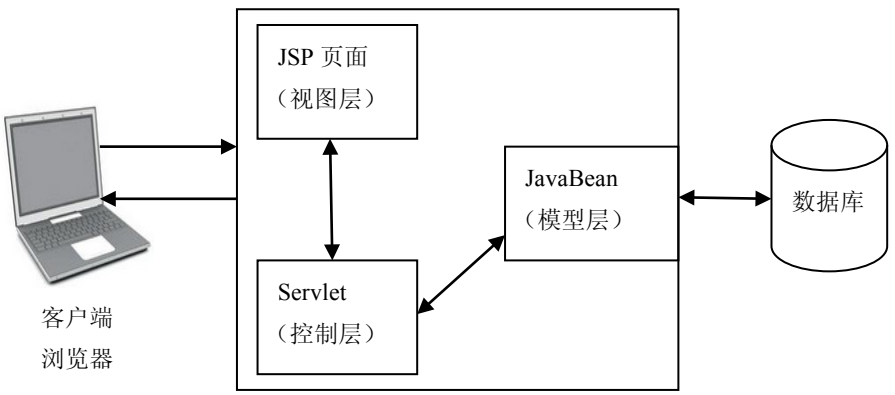


图 4-25　Java Web 中的 MVC

其中：

（1） Model（模型）是 Web 程序中用于表示业务数据和进行业务逻辑操作部分，通常模型对象负责在数据库中存取数据。它是 Web 应用程序的主体，它被调用去处理业务数据，并把处理的结果提供给视图显示。在 MVC 的三个部件中，模型拥有最多的处理任务，被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

（2） View（视图）是应用程序中处理数据显示的部分，通常视图是依据模型数据创建的，是用户在 Web 浏览器中可以看到的界面。视图用来与用户进行交互，显示用户感兴趣的业务数据，并且给用户提供输入界面。视图不进行业务逻辑的处理，而是把用户提交的数据传递到模型中处理，并读取模型处理后的结果去显示。对老式的 Web 应用程序来说，视图就是由 HTML 元素组成的界面，在新式的 Web 应用程序中，HTML 依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，例如：Adobe Flash 和 XHTML，XML/XSL，

WML 等一些标识语言及 Web services。

（3） Controller（控制器）用来接收用户的请求，并调用模型和视图最终完成请求。通常控制器负责从视图读取数据，接受用户的输入并调用模型和视图去完成用户的需求，所以当单击 Web 页面中的超链接或发送 HTML 表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理业务数据，然后再确定用哪个视图来显示返回的数据。

MVC 分层有助于管理复杂的应用程序，开发者可以在一个时间内专门关注一个方面。例如，开发者可以在不依赖业务逻辑的情况下专注于视图设计。同时也让应用程序的测试更加容易。MVC 分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。

MVC 的优点：

（1） 模型可以被重用，在 MVC 设计模式中，视图用来把格式化的数据显示到 Web 浏览器上。而模型则用来处理用户的请求，并返回响应数据。数据表示和业务逻辑相分离，多个视图可以使用同一个模型，同一个模型可以被多个视图去使用。这样就大大提高了用于进行业务逻辑处理的代码的重用性。

（2） 模型是自包含的，它与视图和控制器相独立。因而对应用程序业务逻辑和业务规则的改变不会影响到视图和控制器，这样就带来了很多的方便。例如，想把数据库由 SQL Server 移植到 Oracle，或者想把 RDBMS 数据源改为 LDAP 数据源，则是需要改变模型层即可。一旦实现了模型层，不管业务数据是来自数据库还是来自 LDAP 服务器，视图层都可以正确显示它们。正是由于 MVC 三个模块相互独立，改变其中一个不会影响到另外两个，所以依据这种思想才能构造出松耦合的组件。

（3） 控制器提高了应用程序的灵活性和可配置性。控制器可以连接不同的模型和视图去完成用户的请求，它为构造应用程序提供了强有力的重组手段。一旦给定一些可以重用的视图和模型，控制器可以根据用户的需要选择适当的模型进行业务逻辑的处理，并且选择适当的视图把结果数据显示给用户。

（4） MVC 适合逻辑复杂的大型应用项目，而不适合简单的小型应用程序，对于大型项目，MVC 可以使软件的健壮性、代码的重用性，以及结构的合理性得到很大程度的增强。对于小型项目，或者没有后期开发的项目，MVC 的分层设计会给开发带来额外的工作量。

二、Java Bean

1. JavaBean 简介

JavaBean 是一种 Java 语言写成的可重用组件。为写成 JavaBean，类必须是具体的和公共的，并且具有无参数的构造器。JavaBean 通过提供符合一致性设计模式的公共方法访问成员属性，JavaBean 属性名称符合这种模式，其他 Java 类可以通过自身机制发现和操作这些 JavaBean 属性。换句话说，Javabean 就是一个 Java 的类，只不过这个类要按上面提到的一些规则来写，比如类必须是公共的，无参构造等，JSP 页面可以引用一个或多个

JavaBean 组建对象，同样，一个 JavaBean 对象也可以被多个 JSP 页面所引用，达到了代码重用的效果，提高了开发效率和程序的灵活性。

2. JavaBean 与 Enterprise Bean 的不同

JavaBean 是使用 java.beans 包开发的，它是 Java 2 标准版的一部分。JavaBean 是一台机器上同一个地址空间中运行的组件，JavaBean 是进程内组件。Enterprise Bean 是使用 javax.ejb 包开发的，它是标准 JDK 的扩展，是 Java 2 Enterprise Edition 的一部分。Enterprise Bean 是在多台机器上跨几个地址空间运行的组件。因此 Enterprise Bean 是进程间组件。JavaBean 通常用作 GUI 窗口小部件，而 Enterprise Bean 则用作分布式商业对象。

3. JavaBean 的种类

JavaBean 分为可视化 JavaBean 和非可视化 JavaBean 两种。

（1） 在传统的应用中，JavaBean 主要用于实现一些可视化界面，如一个窗体、按钮和文本框等，这样的 JavaBean 称之为可视化的 JavaBean。

（2） 随着技术的不断发展与项目的需求，目前 JavaBean 主要用于实现一些业务逻辑或封装一些业务对象，由于这样的 JavaBean 并没有可视化的界面，所以又称为非可视化的 JavaBean。

可视化的 JavaBean 一般应用于 Swing 的程序中，在 Java Web 开发中并不会采用，而是使用非可视化的 JavaBean，实现一些业务逻辑或封装一些业务对象。系统中用到的 JavaBean 都是非可视化的 JavaBean。

三、Servlet

1. Servlet 简介

Servlet（Server Applet），全称 Java Servlet，是用 Java 编写的服务器端程序，其实质是按 Servlet 规范编写的 Java 类。与 Java 程序的区别是，Servlet 是一个供其他 Java 程序（Servlet 引擎）调用的 Java 类，它不能独立运行，它的运行完全由 Servlet 引擎来控制和调度。Servlet 对象主要封装了对 HTTP 请求的处理，并且它的运行需要 Servlet 容器的支持。Servlet 是一个标准，它由 Sun 公司定义，其具体细节上的实现由 Servlet 容器进行实现，如 Tomcat、JBoss 等。其主要功能是用来接受、处理客户端请求，并把处理结果返回到客户端显示。其处理过程如下：

（1） 客户端把请求送到 Servlet 处理。

例如：用来实现用户登录的表单，其代码中可以指定使用哪个 Servlet 去处理它，用户填写好表单后，单击“登录”按钮时，客户端将请求发送到 Web 服务器，Web 服务器根据客户端的请求，找到相应的 Servlet 调用它去处理请求。

（2） Servlet 处理请求。

这时 Servlet 相应的方法被 Web 服务器调用去处理请求，这个过程中，Servlet 可以和服务器的其他资源（文件、数据库、Java 应用程序等）交互，例如，该 Servlet 可以调用其他的 Servlet、Java 应用程序。它还可以连接数据库，执行数据库操作，读取文件或者生成文件等。执行完这些操作后，该 Servlet 会把最终客户端想要的结果返回。

（3） Servlet 把请求的结果返回给客户端显示。

在这个过程中，Servlet 返回的并不仅仅是数据结果，而是整个 HTML 文件。例如，用户在网页上查询信息时，假设在单击“查询”按钮后，返回了显示查询结果的页面，那么用来表示这个页面的整个 HTML 文件都是 Servlet 产生并返回来的，而不仅仅是页面上显示的查询结果。

2. Servlet 接口实现类

Servlet 接口 SUN 公司定义了两个默认实现类，分别为：GenericServlet、HttpServlet。HttpServlet 指能够处理 HTTP 请求的 servlet，它在原有 Servlet 接口上添加了一些与 HTTP 协议处理方法，它比 Servlet 接口的功能更为强大。因此开发人员在编写 Servlet 时，通常应继承这个类，而避免直接去实现 Servlet 接口。

HttpServlet 在实现 Servlet 接口时，覆写了 service 方法，该方法体内的代码会自动判断用户的请求方式，如为 GET 请求，则调用 HttpServlet 的 doGet 方法，如为 Post 请求，则调用 doPost 方法。因此，开发人员在编写 Servlet 时，通常只需要覆写 doGet 或 doPost 方法。

3. Servlet 与 JSP 的区别

Servlet 是使用 Java Servlet 接口运行在 Web 服务器上的 Java 程序，其功能十分强大，它不但可以处理 HTTP 请求中的业务逻辑，而且还可以输出 HTML 代码来显示指定页面。而 JSP 是一种在 Servlet 规范智商的动态网页技术，在 JSP 页面之中，同样可以编写业务逻辑处理 HTTP 请求，也可以通过 HTML 代码编辑页面，在实现功能上，Servlet 与 JSP 貌似相同，实际上存在一定的区别，主要有以下几个方面：

（1） 角色不同。

JSP 页面可以存在 HTML 代码与 Java 代码并存的情况，而 Servlet 需要承担客户请求与业务处理的中间角色，只有调用固定的方法才能将动态内容输出为静态的 HTML，所以 JSP 更偏向于显示层的角色。

（2） 编程方法不同。

Servlet 与 JSP 在编程方法上存在很大的区别，使用 Servlet 开发的 Web 应用程序需要遵循 Java 的标准，而 JSP 需要遵循一定脚本语言规范。在 Servlet 代码中，需要调用 Servlet 提供的相关 API 接口方法，才可以对 HTTP 请求及业务进行处理，对于业务逻辑方面的处理功能更加强大。然而在 JSP 页面之中，通过 HTML 代码与 JSP 内置对象实现对 HTTP 请求及页面的处理，其显示界面的功能更加强大。

（3） 运行方式不同。

Servlet 需要在 Java 编译器编译后才可以运行，如果 Servlet 在编写完成或修改后没有被重新编译，则不能在 Web 容器中运行。而 JSP 则与之相反，JSP 由 JSP Container 对其进行管理，它的编辑过程也有 JSP Container 对 JSP 进行自动编辑，所以，无论 JSP 文件被创建还是修改，都不需要对其进行编译就可以执行。

（4） 运行速度不同。

由于 JSP 页面由 JSP Containter 对其进行管理，在每次执行不同内容的动态 JSP 页面时，JSP Containter 都要对其自动编译，所以，它的效率低于 Servlet 的执行效率。而 Servlet 在

编译完成之后，不需要再次编译，可以直接获取及输出动态内容。如果 JSP 页面之中的内容没有变化，JSP 页面的编译完成之后，JSP Containter 就不会再次对 JSP 进行编译了。

4. Servlet 配置

创建了 Servlet 类后，还需要对 Servlet 进行部署，部署的目的是为了将创建的 Servlet 注册到 Servlet 容器中，以方便 Servlet 容器对 Servlet 的调用，部署 Servlet 有两种方式：

（1） 配置 Web.xml 文件。

（2） 注释方式。

下面分别给以介绍：

（1） 配置 Web.xml 文件，其配置方式如下：

```
<servlet>
   <servlet-name>MyServlet</servlet-name>
   <servlet-class>com.Servlet.MyServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>MyServlet</servlet-name>
   <url-pattern>/Servlet</url-pattern>
</servlet-mapping>
```

Servlet 的配置包括两部分：

① <servlet>配置 Servlet 的名字和完整类路径：

servlet-name 是自定义的，就是给 Servlet 取个名字。

② servlet-class 是 Servlet 完整的类，就是从一开始的包一直“.”到该 Servlet。

<servlet-mapping>是用来截获请求的，包括 servlet-name 和 url-pattern。

servlet-name 跟<servlet>中的 servlet-name 是对应的，两个 servlet-name 一定要一致，否则会找不到对应的 Servlet。

url-pattern 是截获请求的规则，当表单提交的时候，会根据特定的规则调用相应的 Servlet。下面会具体阐述。

url-pattern 大致分为以下几种方式：

① 完全匹配。

如：<url-pattern>/servlet/MyServlet.do</url-pattern>

② 目录匹配。

如：<url-pattern>/servlet/*</url-pattern>

③ 扩展名匹配。

如：<url-pattern>*.do</url-pattern>

在 web.xml 文件中，以下语法用于定义映射：

① 以“/”开头和以“/*”结尾的是用来做路径映射的。

② 以前缀“*.”开头的是用来做扩展映射的。

③ “/”是用来定义 default servlet 映射的。

④ 剩下的都是用来定义详细映射的。比如：/aa/bb/cc.action。

容器查找规则：

① 容器会首先查找完全匹配，如果找不到，再查找目录匹配，如果也找不到，就查找扩展名匹配。

② 如果一个请求匹配多个“目录匹配”，容器会选择最长的匹配。

例如：servletA 的 url-pattern 为/test/*，而 servletB 的 url-pattern 为/test/b/*，此时访问 http://localhost/test/b 时，容器会选择路径最长的 servlet 来匹配，也就是这里的 servletB。

注意：“/*.action”这样一个看起来很正常的匹配会错。因为这个匹配既属于路径映射，也属于扩展映射，会导致容器无法判断。

如果使用“Eclipse IDE for JavaEE”或“MyEclipse”等开发工具提供的开发界面创建 Servlet，可以省略 Servlet 的配置步骤，因为开发工具会自动配置。

（2） 加注释方式。

@WebServlet 用于将一个类声明为 Servlet，该注解将会在部署时被容器处理，容器将根据具体的属性配置将相应的类部署为 Servlet。

在“public class MyServlet extends HttpServlet {”代码前加入如下注释

```
@WebServlet("/MyServlet")
```

如果使用第一个模块介绍的新版本的“Eclipse IDE for JavaEE”开发工具提供的开发界面创建 Servlet，开发工具会自动通过注释配置 Servlet，所以，用户不需要自己配置。

习题

1. 简述 MVC 设计模式。
2. 存储过程的优点有哪些？
3. 游标的使用顺序是什么？

实训项目

1. 创建一存储过程，返回学生的姓名，选修的课程名和成绩。
2. 创建一存储过程，实现输入学生姓名返回学生的基本信息。
3. 创建一存储过程，输入学生的学号，返回学生选修门数和平均分。
4. 创建一存储过程，实现把女生人数超过 600 人的系查找处理，并把系的名称，女生人数插入到一张 girlcount 表中。
5. 完善学生成绩管理系统，实现管理员用户对学生信息的查找、插入、删除和修改功能，及对成绩和课程信息的查找功能，查找参考案例 1，实现分类查找，字符型数据可以按关键字精确查找、模糊查找。数字型数据实现按关键字精确查找、范围查找、集合查找和条件查找。
6. 完善学生成绩管理系统，实现学生用户对个人成绩信息的查找功能和课程信息的查找功能，查找参考案例 1，实现分类查找，字符型数据可以按关键字精确查找、模糊查找。数字型数据实现按关键字精确查找、范围查找、集合查找和条件查找。

案例 2　用户密码加密

任务 1　添加注册页面

在前面设计的登录页面上添加一个注册按钮，用于用户注册，当用户单击注册按钮时会弹出注册窗口。具体操作步骤如下：

（1） 修改 login.jsp 代码，在“<input type="reset" style='font-size: 14px' value="清空">”代码后面加入代码：“   <input type="button" style='font-size: 14px' value="注册" onClick="window.location.href='regist.jsp'" >”。

（2） 建立 regist.jsp 页面代码如下：

```
<%@ page language="java" pageEncoding="UTF-8"%>
<title>注册页面</title>
<head>
</head>
<script type="text/javascript">
    function checksubmit() {
           with (document.logform) {
                  var uid = userid.value;
                  var pwd = upass.value;
                  if (uid == null || uid == "") {
                         alert("请填写账号 !");
                         return;//函数返回
                  } else if (pwd == null || pwd == "") {
                         alert("请填写密码!");
                         return;
                  } else
                         document.logform.submit();
           }
    }
</script>
<body>
    <H2 align="center">请注册</H2>
    <hr>
    <form action="Regist" method="post" name="registform">
           <table border="1" class="font" align="center">
                  <tr>
                         <td align="center">用户账号: </td>
    <td><input type="text" name="userid" style="width: 160px" /></td>
                  </tr>
                  <tr>
    <td align="center">密&amp;nbsp;&amp;nbsp;码: </td>
<td><input type="password" name="userpass" style="width: 160px" /></td>
                  </tr>
                  <tr>
<td align="center"><input type="submit" value="注册"
```

```
            style='font-size: 14px' onClick="checksubmit()" /></td>
    <td align="center"><input type="reset" style='font-size: 14px'
                                    value="清空"></td>
                    </tr>
            </table>
        </form>
    </body>
    </html>
```

任务 2　建立加密类

Java 中提供了专门的 MD5 工具类，java.security.MessageDigest 类提供的专门用于 MD5 单项数据加密的方法，可以用于对数据库中的数据进行 MD5 加密。下面以教师用户注册为例介绍采用 MD5 加密算法，实现对用户密码进行加密后存入数据库，登录时把读取用户输入的登录密码进行 MD5 加密，然后和从后台数据库读取的该用户 MD5 加密后的密码进行比较，如果两者相同，则成功登录，如果不相同，则表示密码有误，登录失败。建立 MD5 加密类的步骤如下。

建立 com.ssms.bean 包，在该包中建立名为 MD5Encode 的类，具体代码如下：

```
package com.ssms.bean;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
 public final class MD5Encode {
    private MD5Encode() {
     }
    /**
/* 获取字符串 MD5 码，系统默认编码格式
* @param source
* @return
*/
    public static String getMD5(String source) {
        return MD5Encode.getMD5(source.getBytes());
    }
     /**
     * 获取字符串 MD5 码，指定编码格式
     *
     * @param source
     * @param charset
     * @return
     */
    public static String getMD5(String source, String charset) {
        String output = null;
        try {
            output = MD5Encode.getMD5(source.getBytes(charset));
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
```

```
        return output;
    }

    /**
     * 获取字节数组 MD5 码
     *
     * @param in
     * @return
     */
    public static String getMD5(byte[] in) {
        String result = null;
        try {
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            StringBuilder sbu = new StringBuilder();
            byte[] out = md5.digest(in);
            for (byte b : out) {
                int o = b & 0xFF;
                if (o < 16) {
                    sbu.append("0");
                }
                sbu.append(Integer.toHexString(o));
            }
            result = sbu.toString().toUpperCase();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return result;
    }
}
```

任务 3　完成注册用户密码加密

（1） 在 TeacherDao 接口中增加如下一行代码：

```
    public boolean isRegist(String  tid,String tpws);
```

（2） 在 TeacherDaoImpl 类中增加如下代码：

```
public boolean isRegist(String id,String pws){
          Connection conn=null;
          PreparedStatement pstmt=null;
          boolean ret=false;
      conn=BaseDao.getConn();
        String sql="insert into teacher (tid,tpassword) values(?,?)";
           try {
          pstmt=conn.prepareStatement(sql);
          pstmt.setString(1,id);
          pstmt.setString(2, pws);
          int n=pstmt.executeUpdate();
          if(n!=0){ret=true;}
```

```
        } catch (SQLException e) {
              e.printStackTrace();
        } finally {
              BaseDao.closeAll(null, pstmt, conn);
        }
                    return ret;
        }
```

同时修改 TeacherDaoImpl 类中的 IsLogin 方法中的代码，在“conn=BaseDao.getConn();”代码前加入下面一行代码：

```
    tpws=MD5Encode.getMD5(tpws);
```

（3） 在 TeacherBiz 接口中增加如下一行代码：

```
       public boolean isRegist(String tid,String tpws);
```

（4） 在 TeacherBizImpl 类中增加如下代码：

```
    public boolean isRegist(String tid, String tpws) {
              return tdao.isRegist(tid, tpws);
       }
```

（5） 建立名为 Regist 的 servlet，其代码如下：

```
    package com.ssms.server;
    import java.io.IOException;
    import java.io.PrintWriter;
    import javax.servlet.ServletException;
    import javax.servlet.annotation.WebServlet;
    import javax.servlet.http.HttpServlet;
    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;
    import com.ssms.bean.MD5Encode;
    import com.ssms.biz.TeacherBiz;
    import com.ssms.biz.impl.TeacherBizImpl;
    @WebServlet("/Regist")
    public class Regist extends HttpServlet {
       private static final long serialVersionUID = 1L;
       public Regist() {
              super();
       }
       protected void doGet(HttpServletRequest request, HttpServletResponse
response)
                    throws ServletException, IOException {
              doPost(request, response);
       }
       protected void doPost(HttpServletRequest request, HttpServletResponse
response)
                    throws ServletException, IOException {
              request.setCharacterEncoding("UTF-8");
```

```
            response.setCharacterEncoding("UTF-8");
            response.setContentType("text/html;charset=UTF-8");
            String uid = request.getParameter("userid");
            String upws = request.getParameter("userpass");
            upws = MD5Encode.getMD5(upws);
            boolean ret = false;
            TeacherBiz tbiz = new TeacherBizImpl();
            PrintWriter out = response.getWriter();
            ret = tbiz.isRegist(uid, upws);
            if (ret) {
                  out.println("<script>alert('成功注册');window.location.href
='login.jsp';</script>");
            } else {
                  out.println("<script>alert('注册失败！');window.location.
href ='regist.jsp';</script>");
            }
      }
   }
```

（6） 修改 TeacherDaoImpl 类中 isLogin 方法。

（7） 加密后存储在数据库中的数据是加密后的数据，所以用户登录时，不能直接把输入的数据和数据库读取的数据进行比较验证登录，用户输入的数据需要用 MD5 加密后，再和数据库读取的数据进行比较验证登录，才能正确登录，所以验证加密后用户的登录需要修改 TeacherDaoImpl 类中 isLogin 方法，在代码“String sql="select * from teacher where tid=? and tpassword=?";”，前加入代码：“tpws=MD5Encode.getMD5(tpws);”，用户修改密码部分的代码在存入数据库之前也要进行加密，保证数据库中用户的密码都是加密的。

下面运行项目，注册一个用户，查看一下加密效果。

运行项目在弹出的如图 4-26 的登录页面中，单击注册按钮会弹出如图 4-27 所示的注册页面，假设输入用户名 20070788，密码也为 20070788，单击注册按钮，会弹出如图 4-28 所示的注册成功消息框。这样注册的用户信息已经写入数据库，密码是经过 MD5 加密后的效果，通过运行 SQL Server Management Studio Express，新建查询输入指令“select tId,tPassword from teacher where tId='20070788'”，查询上面注册的用户信息，会得到如图 4-29 所示的结果，可以看到密码是加密后效果。

图 4-26　登录窗口

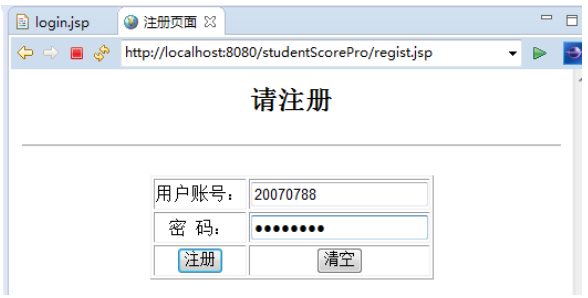


图 4-27　注册窗口

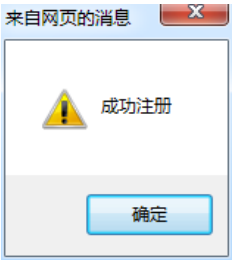


图 4-28　注册成功消息框

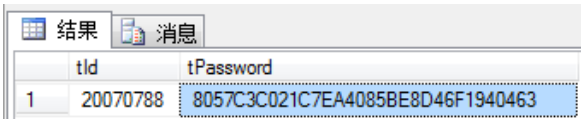


图 4-29　密码加密后存储在数据库中的结果

相关知识

MD5 加密算法

数据加密，是一门历史悠久的技术，数据加密的过程就是对原来为明文的文件或数据按某种算法和加密密钥进行处理，使其成为不可读的一段代码，通常称为密文，通过这种方式达到保护用户数据不被非法窃取、阅读的目的。加密的逆过程是解密，即将密文通过解密算法和解密密钥转化为明文。数据加密的核心是密码学。数据加密目前仍是计算机系统对信息进行保护的一种最可靠的办法。它利用密码技术对信息进行加密，实现信息隐蔽，从而起到保护信息安全的作用。

在现阶段，有两种加密方式：双向加密和单向加密。双向加密是加密算法中最常用的，它将可以直接理解的明文数据加密为不可直接理解的密文数据，然后，在需要的时候，可以使用一定的算法将这些加密以后的密文解密为原来可以阅读的明文。双向加密适合于隐秘通信，例如，军区之间互通信息时，当然不希望传送的数据敌方能阅读或者理解，因此，在信息传送前一定对传送的信息加密，接收方收到信息后，通过解密算法就可以得到准确的信息。单向加密刚好相反，只能对数据进行研究加密，也就是说，没有办法对加密以后的数据进行解密。单向加密一般用于数据库中用户信息的加密，如当用户在某网站注册成会员时，他的账号及密码不是直接保存到数据库中，而是经过一次单向加密后再保存，这样，即使这些信息被泄露，也不能立即理解这些信息的真正含义。

MD5 算法的中文名称是信息-摘要算法（英文全称是 Message-Digest Algorithm 5），是计算机广泛使用的杂凑算法之一，是一种单向加密算法，该算法广泛应用于互联网网站的用户文件加密，能够将用户密码加密为 128 位的长整数。在 20 世纪 90 年代初由 MIT Laboratory for Computer Science and RSA Data Security Inc 的 Ronald L. Rivest 开发出来，经 MD2、MD3 和 MD4 发展而来。

MD5 算法有两个重要特性：第一是任意两段明文数据，加密以后的密文是不相同的；

第二是任意一段明文数据，经过加密以后，得到的密文一定是相同的。

MD5 的作用是让大容量信息在用数字签名软件签署私人密钥前被“压缩”成一种保密的格式（就是把一个任意长度的字节串变换成一定长的十六进制数字串），当前主流编程语言普遍已有 MD5 实现方法。

习题

1. 目前常用的加密方式有哪些？
2. 什么是 MD5 加密算法？
3. MD5 加密算法的重要特性是什么？

实训项目

1. 完成注册用户的加密。
2. 完善系统，对学生用户和管理员用户实现密码修改功能，实现修改后的密码存入数据库要进行加密。

案例 3　成绩管理系统的安全控制

任务 1　创建学生成绩管理系统数据库用户

用户要访问 SQL Server 必须先登录到 SQL Server 服务器，用户要登录到服务器，就需要有一个登录账户，用户登录时，SQL Server 服务器要对登录账户进行合法性检查，即身份验证，只有身份验证通过后才能登录服务器。SQL Server 对身份验证有两种方式，分别是 Windows 身份验证和 SQL Server 身份验证，有关这两种身份验证的详细介绍请参考案例后的相关知识部分。只有合法的登录用户才能成为数据库的用户，所以对每个数据库用户首先为其创建合法的登录账户，然后再把对应的登录账户映射为数据库用户。下面分别从交互方式和命令方式两个方面介绍 SQL Server 用户的创建和删除过程。

一、交互方式下用户的创建

创建登录账户的操作只有管理员才可以使用，所以要建立登录账户，必须以管理员身份，打开 SQL Server Management Studio，连接到 SQL Server 服务器。

1.　创建 SQL Server 登录账号

【例】为学号为 1234120101 的学生，建立 SQL Server 登录账户，用户名密码均为该生学号。

交互方式下创建登录帐号方法是：以超级用户的身份登录 SQL Server 服务器，打开 Microsoft SQL Server Management Studio →对象资源管理器→安全性→登录名下单击鼠标右键选择新建登录名如图 4-30 所示，然后会弹出如图 4-31 所示对话框。为安全起见这里选择 SQL Server 身份验证，输入用户名和密码，去掉强制密码过期选项，默认数据库选择学生成绩管理系统数据库 ssms，如图 4-31 所示。

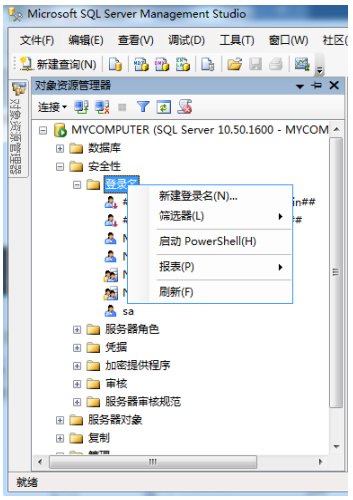


图 4-30　新建 SQL Server 登录账号

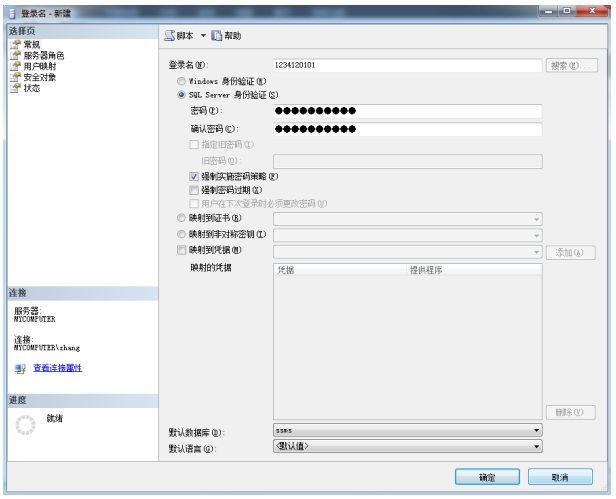


图 4-31　新建 SQL Server 登录账号窗口

2.　把 SQL Server 的登录用户变成成绩管理系统 ssms 的数据库用户

【例】把学号为 1234120101 的学生，建立成绩管理系统 ssms 数据库用户，用户名密码为该生学号，默认数据库为学生成绩管理数据库 ssms。

打开 Microsoft SQL Server Management Studio，对象资源管理器→数据库→ssms→安全性→用户→单击鼠标右键→新建用户如图 4-32 所示，会弹出如图 4-33 所示对话框。输入用户名，然后单击登录名右边的命令按钮，会弹出如图 4-34 所示窗口，因为在 SQL Server 中，数据库用户和登录用户具有对应关系，一个数据库用户必须是一个合法的数据库登录用户，一个数据库登录用户可以映射为多个数据库用户。

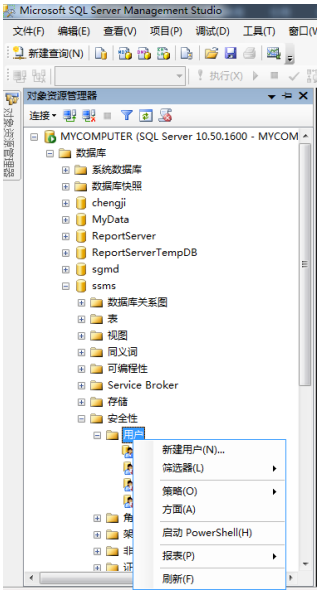


图 4-32　新建数据库用户操作

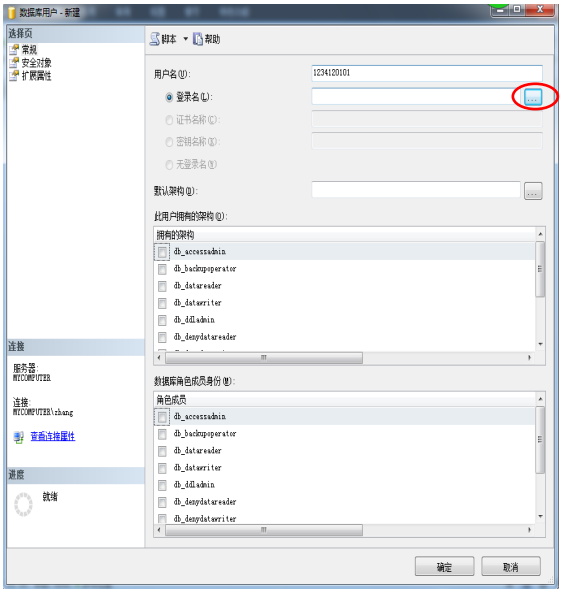


图 4-33　新建数据库用户窗口

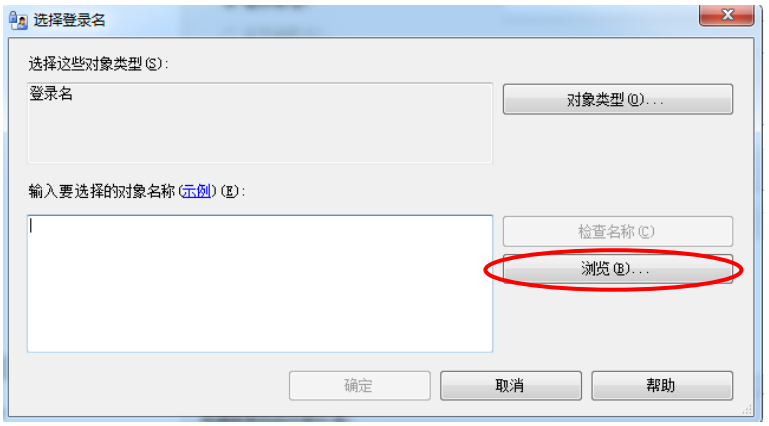


图 4-34　新建用户中选择对应登录用户

在图 4-34 中单击浏览按钮，会弹出如图 4-35 所示的查找对象窗口，选择对应的登录用户，然后单击所有弹出窗口的确定按钮，则完成了数据库用户的创建，在学生成绩管理数据库 ssms 的安全性的用户下面可以看到创建的用户信息，如图 4-36 所示。

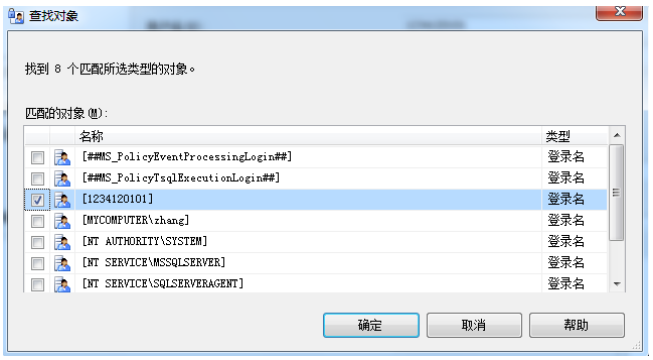


图 4-35　新建用户选择对应登录用户

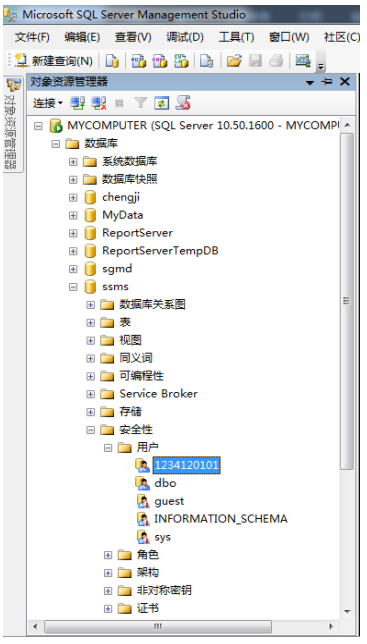


图 4-36　查看已经建立的数据库用户

3.　删除 SQL Server 登录用户

删除 SQL Server 登录用户前，先删除此登录用户在数据库上映射的数据库用户，如果没有删除登录用户映射的数据库用户，而直接删除登录用户会弹出如图 4-37 所示的错误提示信息。

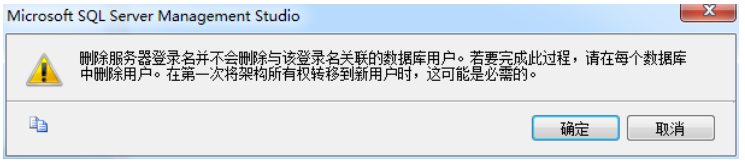


图 4-37　删除登录用户错误提示窗口

所以在删除登录用户前先删除登录用户映射的数据库用户具体操作步骤是：

打开 Microsoft SQL Server Management Studio，“对象资源管理器”→“数据库”→“ssms”→“安全性”→“用户”→找到要删除的用户，单击鼠标右键选择其中的删除选项如图 4-38 所示。

删除了登录用户映射的数据库用户后，即可删除该登录用户，具体操作步骤是：

打开 Microsoft SQL Server Management Studio，“对象资源管理器”→“数据库”→“ssms”→“安全性”→“用户”→登录名中找到要删除的登录用户，单击鼠标右键选择其中的删除选项，即删除了选择的登录用户，如图 4-39 所示。

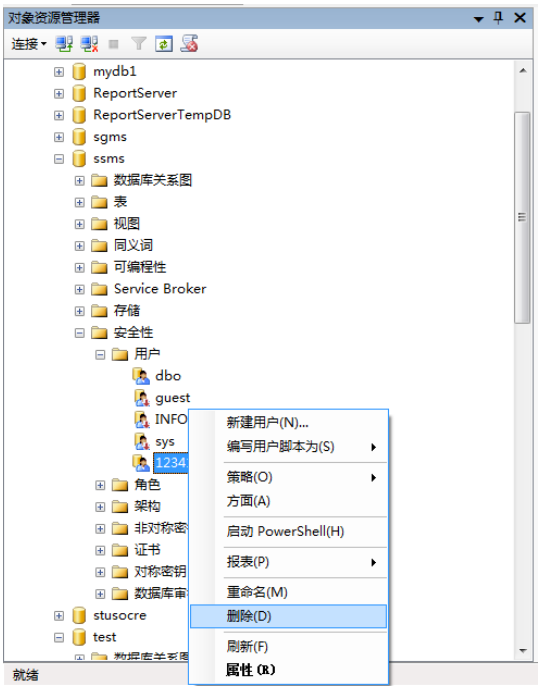


图 4-38　删除数据库用户

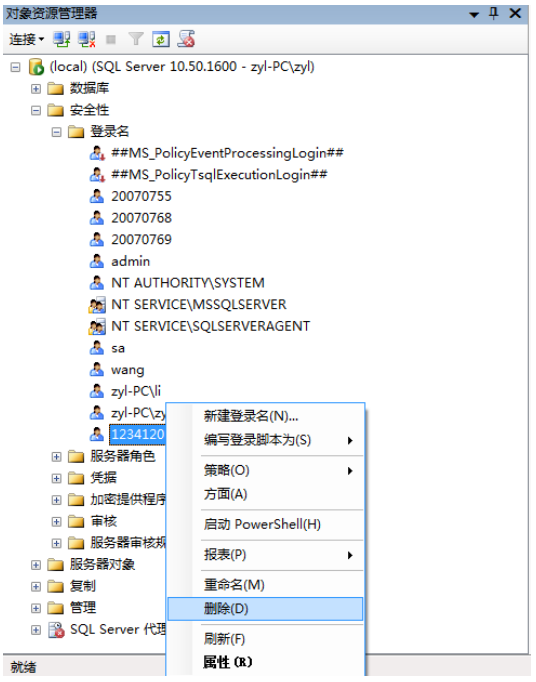


图 4-39　删除 SQL Server 登录账号

4.　建立 Windows 登录用户

【例】为 Windows 登录用户 1234120102，建立 Windows 认证的登录账户，帐号和密码均为该生学号。

首先在 Windows 操作系统为该生建立 Windows 系统的登录账户。下面以 Win7 环境为例介绍创建 Win7 系统的登录用户，具体操作步骤如下：

在桌面上的“计算机”上单击鼠标右键，选择其中的“管理”菜单项，如图 4-40 所示，会弹出图 4-41 的计算机管理窗口，在其中的“本地用户和组”中的用户选项上单击鼠标右键选择“新用户”，会弹出图 4-42 的新用户窗口，输入相应的用户名和密码，去掉用户下次登录时须更改密码选项，然后单击“创建”，则 Windows 系统登录账户创建完毕。

图 4-40　单击计算机右键快捷菜单

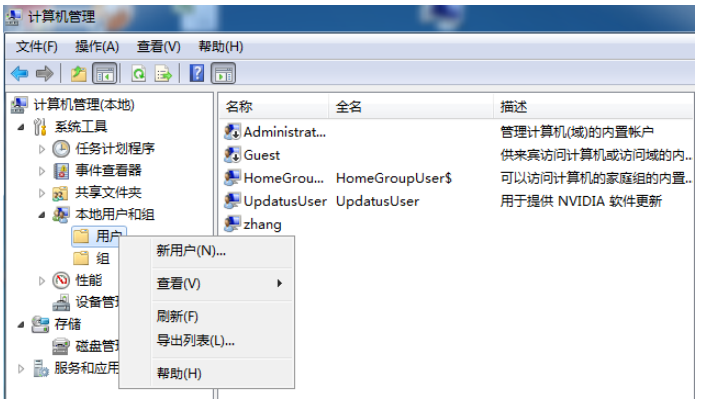


图 4-41　计算机管理窗口

图 4-42　新建操作系统登录用户窗口

5.　创建 SQL Server 中 Windows 身份验证的登录用户

Windows 操作系统的管理员或超级用户可以直接登录 SQL Server，但是 Windows 的普通用户必须由数据库管理员为其创建对应的 SQL Server 登录用户才可以登录 SQL Server。下面为 Windows 登录账户 1234120102 创建 SQL Server 中 Windows 身份验证的登录用户。用户名为 1234120102，默认数据库为成绩管理系统数据库 ssms。具体操作步骤如下：在 SQL Server 的对象资源管理器中的“安全性”中的“登录名”上单击鼠标右键选择“新建登录名”，如图 4-43 所示，然后弹出如图 4-44 所示的新建登录用户窗口，在图 4-44 所示

窗口中的登录名上输入登录 SQL Server 对应的用户名 1234120102，单击“检索”按钮查找对应的 Windows 用户名，会弹出如图 4-45 所示的选择用户或组窗口，单击其中的“高级”按钮，会弹出如图 4-46 所示的选择用户或组窗口，单击其中的“立即查找”，在检索结果中会显示系统当前的所有登录账户，如图 4-47 所示，选择刚才建立的 Windows 登录账户 1234120102，然后逐个单击打开窗口的确定按钮，关闭打开的窗口，在关闭如图 4-48 所示的创建登录用户窗口前修改用户登录的默认数据库为 ssms，最后单击图 4-48 所示窗口的“确定”按钮，完成了 Windows 身份验证的登录用户的创建。

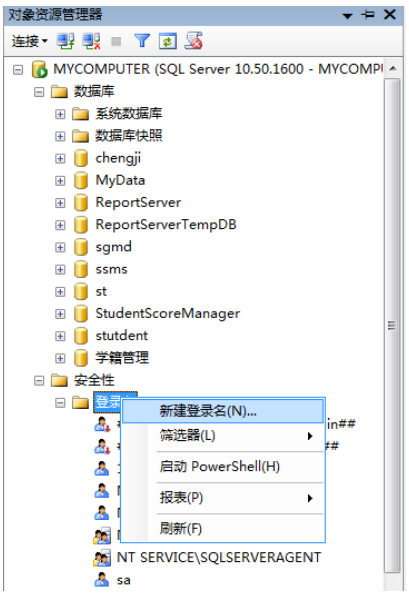


图 4-43　新建登录用户

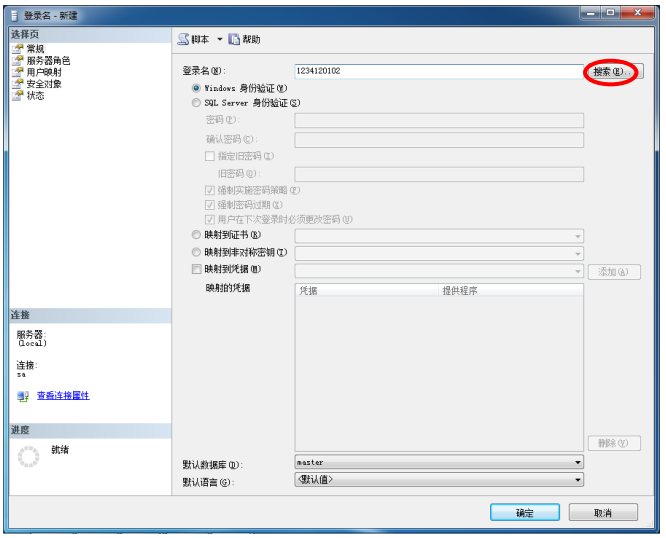


图 4-44　新建 Windows 登录用户窗口

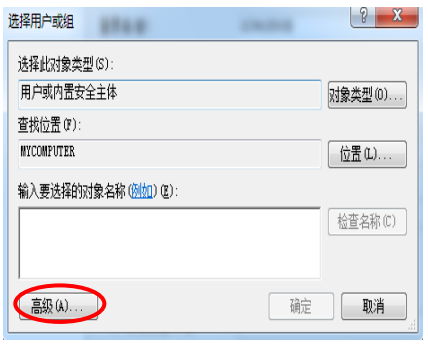


图 4-45　选择用户或组

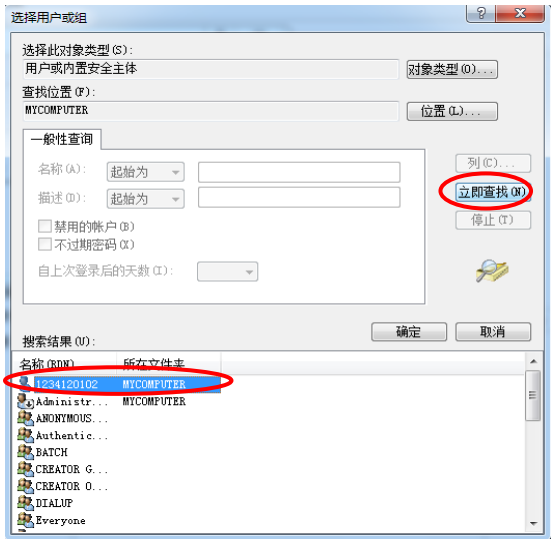


图 4-46　选择用户或组

图 4-47　选择用户或组

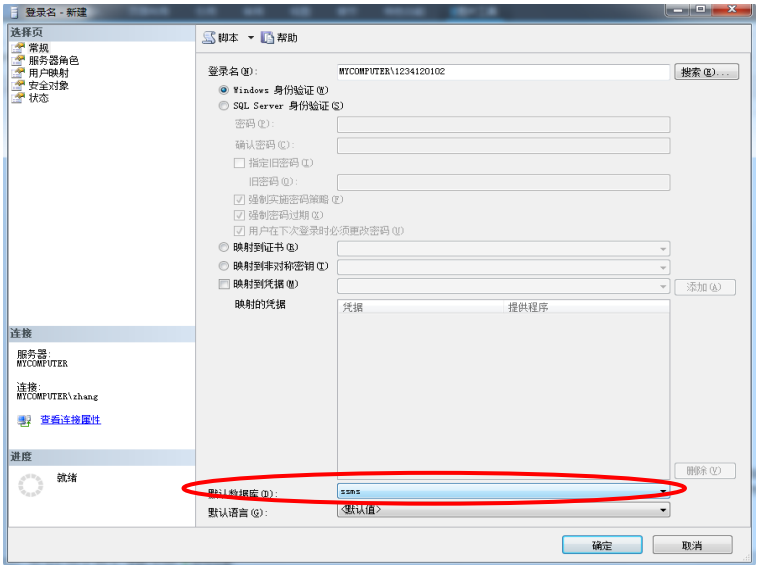


图 4-48　新建 Window 登录用户

6.　把 Windows 身份验证的登录用户变成成绩管理系统 ssms 的数据库用户

把 Windows 身份验证的登录用户变成数据库用户和把 SQL Server 身份验证的登录用户变成数据库用户的步骤和方法完全相同，这里不再赘述。

7.　删除 Windows 身份验证的登录用户

删除 Windows 身份验证的登录用户，与删除 SQL Server 登录方法和步骤相同，需要注意的是，先删除此登录用户映射的数据库用户，然后再删除该登录用户。这里不再赘述。

二、用系统存储过程创建数据库用户

创建登录账户的指令只有管理员才可以使用，所以要建立登录账户，必须以管理员身份，打开 SQL Server Management Studio，连接到 SQL Server 服务器。

1.　用系统存储过程创建 SQL Server 身份验证的数据库登录账户

SQL Server 的存储过程 sp_addlogin 用于创建 SQL Server 身份验证的登录账户，其语

法是：

```
sp_addlogin  '登录账户名', '登录密码', '默认数据库', '默认语言'
```

说明：

（1） “默认数据库”是在登录后首先连接到该数据库，默认值为 master;

（2） 如果不指定“默认语言”，则使用服务器当前的默认语言。

【例】为学号为 1234120103 的学生创建 SQL Server 的登录帐号，用户名和密码均为为该生学号，默认数据库为学生成绩管理数据库 ssms。以超级用户的身份登录 SQL Server 服务器，新建查询，输入如下指令：

```
sp_addlogin '1234120103' ,'1234120103','ssms'
```

2. 用系统存储过程为 SQL Server 的登录账户创建映射的数据库用户

SQL Server 的存储过程 sp_grantdbaccess 用于将 SQL Server 的登录账户映射为数据库用户，其语法是：

```
sp_grantdbaccess '登录账户名','数据库用户名'
```

说明：

（1） 登录账户名可以使映射到新数据库用户的 Windows 组、Windows 登录名或 SQL Server 登录名的名称。Windows 组和 Windows 登录名的名称必须按“域\登录名”的形式（例如，LONDON\Joeb），以 Windows 域名进行限定。如果数据库安装在本机上面，“Windows 域名”对应的是“完整的计算机名称”，在“系统属性”的“计算机名称”中可以看到。登录账户名不得是已映射到数据库中的用户。

（2） 数据库用户名可以省略，如果省略，则默认为登录名。

例如：把上面创建的 SQL Server 登录用户 1234120103 映射为成绩管理系统 ssms 数据库的用户。首先以超级用户或成绩管理系统数据库的所有者的身份连接 SQL Server 服务器，新建查询，把当前数据库切换到成绩管理系统数据库 ssms，输入如下指令：

```
sp_grantdbaccess '1234120103','1234120103';
```

3. 用系统存储过程创建 Windows 身份验证的登录用户

SQL Server 的存储过程 sp_grantlogin 用于创建 Windows 身份验证的登录用户，其语法是：

```
sp_grantlogin '登录账户名'
```

说明：

“登录账户名”必须使用以下格式用 Windows 域名限定 Windows 用户或用户组，例如 Domain\User。

在创建 Windows 身份验证的数据库用户前，首先要在操作系统中创建一个 Windows 系统的登录用户。假设在系统中已经创建了一个用户名为 1234120104 的用户，计算机名称为 mycomputer，则创建 Windows 身份验证的登录用户指令为：

```
sp_grantlogin [mycomputer\1234120104]
```

4. 用系统存储过程为 Windows 身份验证的登录用户创建对应的数据库用户

用系统存储过程为 Windows 身份验证的登录用户创建映射的数据库用户，其语法同为 SQL Server 登录账户创建映射的数据库用户操作。例如把上面创建的 1234120104 的 Windows 登录账户变为数据库用户，用户名为 1234120104，指令如下：

```
sp_grantdbaccess [mycomputer\1234120104],[1234120104]
```

5. 用系统存储过程删除数据库用户

SQL Server 的存储过程 sp_revokedbaccess 用于删除数据库用户，其语法是：

```
sp_revokedbaccess  '数据库用户名'
```

说明：

'数据库用户名' 是要删除的数据库用户名称。

例如删除前面创建的数据库用户 sp_revokedbaccess '1234120101'。

说明：

在 SQL Server 2000 中删除数据库用户很简单——只要直接删除就可以了，但是 SQL Server 2005 以后版本中直接删除是不行的，这是因为在 SQL Server 2005 以后版本架构是作为实体对待的，新建立的用户即拥有了一个和其数据库用户名相同的架构。所以要删除该用户需要先删除该用户所拥有的架构或更改此架构的所有者。否则会弹出如图 4-49 所示的错误提示。

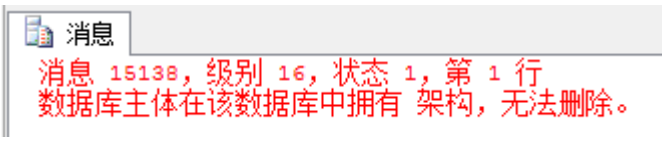


图 4-49　删除数据库用户错误提示窗口

每一个数据库用户对应于一个系统登录帐号，并且每一个架构只能赋予一个用户，所以直接删除用户的架构是不可以的，但是，一个用户可以拥有多个架构，在删除一个用户之前先把该用户拥有的架构修改给其他用户才能删除该用户。例如要直接删除前面建立的数据库用户“1234120104”出错，通过下面的指令进行架构转移给 dbo 用户，才可以删除该数据库用户“1234120104”。

```
Alter  AUTHORIZATION  ON  SCHEMA::[1234120104]  TO  dbo;
```

6. 用系统存储过程删除 SQL Server 认证的登录用户

SQL Server 的存储过程 sp_droplogin 用于删除 SQL Server 认证的登录用户，其语法是：

```
sp_droplogin '登录账户名'
```

说明：

“登录账户名”是要删除的登录名。

例如，删除前面建立的 SQL Server 登录账户 1234120103：

```
sp_droplogin '1234120103'
```

7. 用指令删除 Windows 认证的登录用户

SQL Server 的存储过程 sp_revokelogin 用于删除 Windows 认证的登录用户，其语法是：

```
sp_revokelogin  '登录账户名'
```

参数说明：

“登录账户名”是 Windows 用户或组的名称。

例如，删除前面建立的 Windows 登录账户[mycomputer\1234120104]：

```
sp_revokelogin  [mycomputer\1234120104]
```

任务 2 创建学生成绩管理系统数据库学生角色

学生成绩管理系统有多个用户，不同用户有不同的权限，但是部分用户对某些对象具有相同的权限，为了管理方便，可以把具有相同权限的用户创建一个角色，通过对角色授权，具有该角色的用户具有该角色的相关权限，可以简化权限管理。SQL Server 提供了固定服务器角色（8 种，如 sysadmin）和固定数据库角色（10 种，如 db_datawriter），此外，用户还可以创建数据库角色。有关 SQL Server 固定服务器角色和固定数据库角色由于篇幅有限，这里不再详细介绍，读者可以查阅相关资料。在学生成绩管理系统中创建四种类型的角色：管理员、教师、教务员和学生。

下面分别从交互方式和指令形式介绍角色的创建。

一、交互方式创建角色

创建角色指令只有管理员才可以使用，所以要建立角色，必须以管理员身份，打开 SQL Server Management Studio，连接到 SQL Server 服务器。

1. 交互方式下角色的创建

下面以管理员角色 admin 的创建为例，介绍交互方式下创建角色的方法，打开 Microsoft SQL Server Management Studio，对象资源管理器→数据库→ssms→安全性→角色→数据库角色→单击鼠标右键→新建数据库角色如图 4-50 所示。

然后会弹出如图 4-51 所示的新建数据库角色窗口，在角色名称框中输入用户角色名 admin，然后单击确定按钮，完成 admin 角色的创建，以同样的方式，建立教师角色 teacher、教务员角色 acdemicDean 和学生角色 student。这三种角色的交互方式创建这里不再赘述，请读者自己完成。

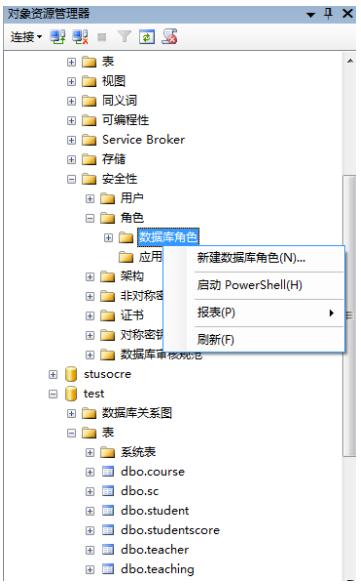


图 4-50　新建数据库角色

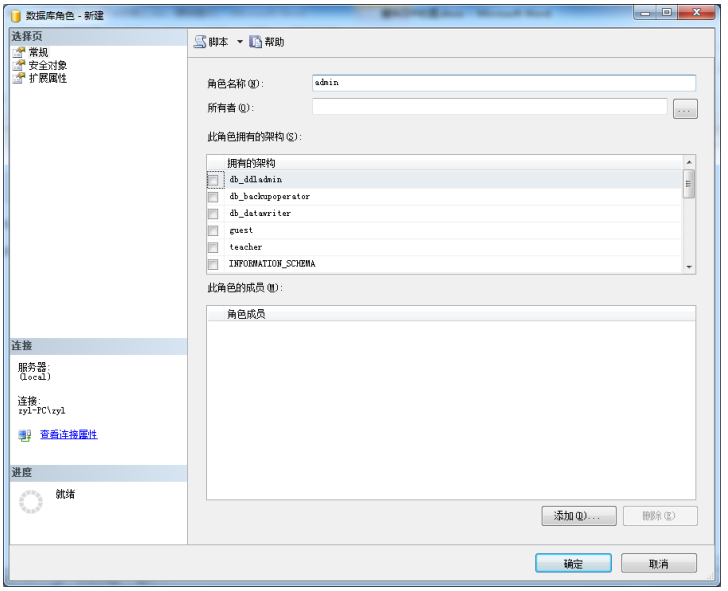


图 4-51　新建数据库角色

2.　交互方式下为角色中添加成员

下面以向学生 student 角色添加一个用户名为 1234120102 的学生为例，介绍交互方式下向角色添加成员的方法，具体操作步骤如下：打开 Microsoft SQL Server Management Studio，对象资源管理器→数据库→ssms→安全性→角色→单击鼠标右键选择“属性”菜单项如图 4-52 所示，然后会弹出如图 4-53 所示的角色属性窗口，单击“添加”按钮，会弹出如图 4-54 所示的选择数据库用户或角色窗口，单击其中的“对象类型”按钮，会弹出如图 4-55 所示的选择对象类型窗口，去掉“数据库角色”复选框，然后单击“确定”按钮，又回到图 4-54 所示窗口，单击其中的“浏览”按钮，然后会弹出如图 4-56 所示的查找对象窗口，依次单击弹开窗口的“确定”按钮，即完成了角色用户的添加。

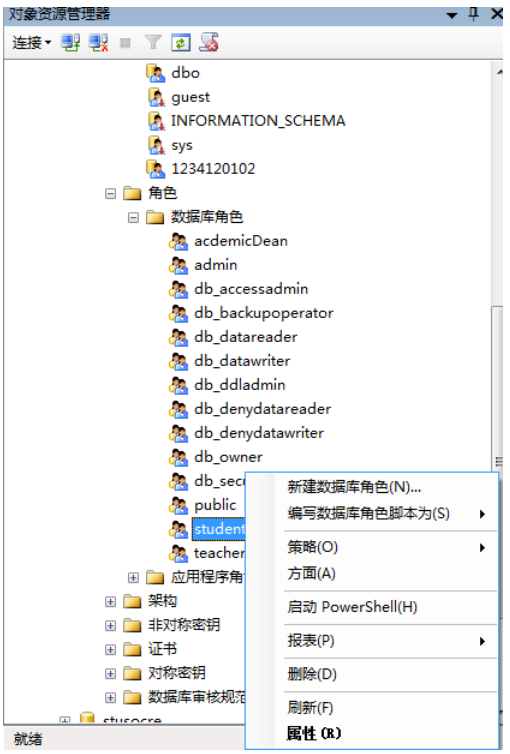


图 4-52　为角色添加成员

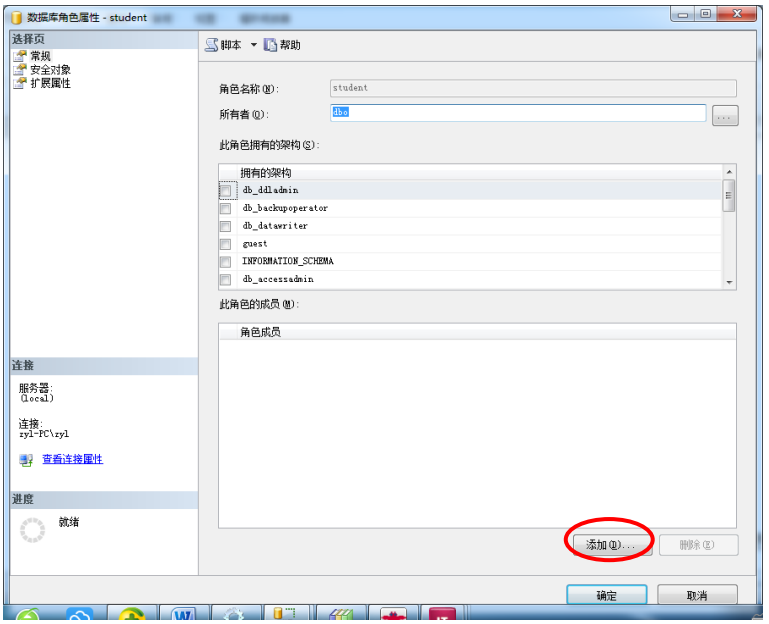


图 4-53　角色属性显示窗口

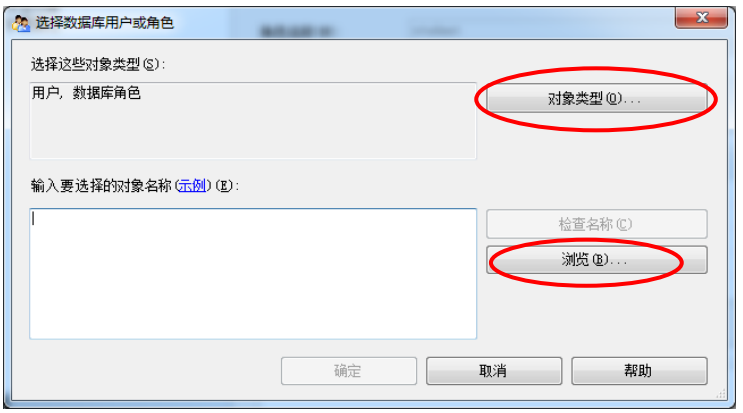


图 4-54　选择数据库用户或角色

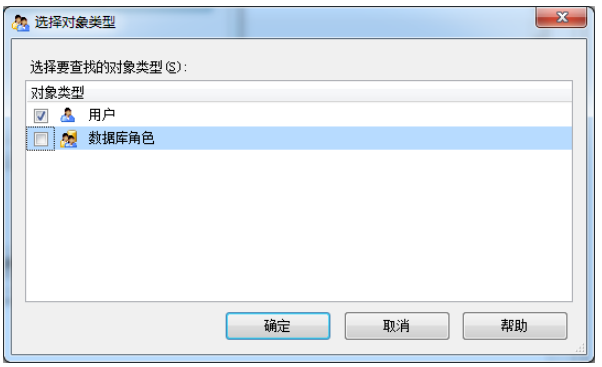


图 4-55　选择对象类型

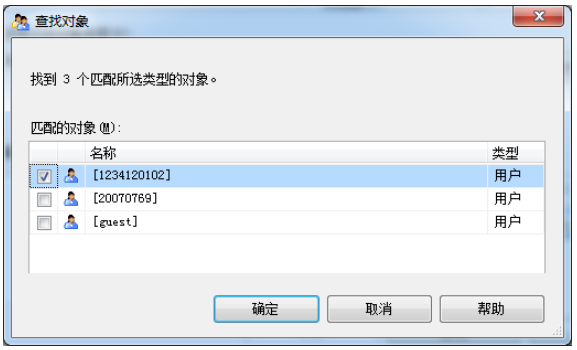


图 4-56　查找对象

3.　交互方式取消角色成员

下面以取消学生 student 角色中名为 1234120102 的成员为例，介绍在交互方式下取消角色成员的方法 ，具体操作步骤如下：打开 Microsoft SQL Server Management Studio，对象资源管理器→数据库→ssms→安全性→角色→单击鼠标右键选择“属性”菜单项，然后会弹出如图 4-57 所示的角色属性窗口，单击“删除”按钮完成取消角色成员的操作。

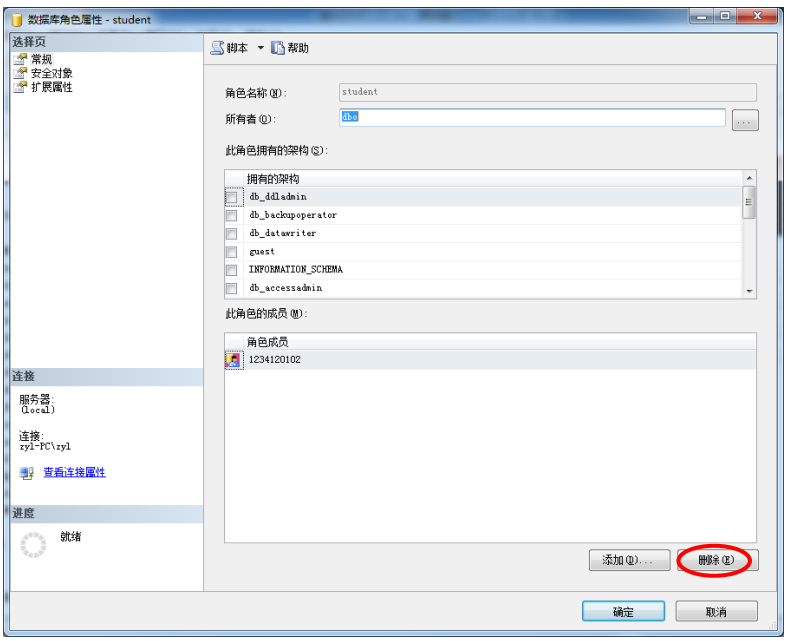


图 4-57　取消角色成员

二、指令形式创建角色

创建角色指令只有管理员才可以使用，所以要建立角色，必须以管理员身份，打开 SQL Server Management Studio，连接到 SQL Server 服务器。

1. 创建用户角色

SQL Server 的存储过程 sp_addrole 用于创建角色，其语法是：

```
sp_addrole  '数据库角色名'
```

说明：

“数据库角色名”是新建立的数据库角色的名称。

【例】用存储过程为学生成绩管理系统中创建四种类型的角色：管理员、教师、教务员和学生。

```
sp_addrole   'admin'
sp_addrole   'teacher'
sp_addrole   'acdemicDean'
sp_addrole   'student'
```

2. 删除角色

SQL Server 的系统存储过程 sp_droprole 用于删除数据库角色，其语法是：

```
sp_droprole '数据库角色名'
```

说明：

“数据库角色名”是要从当前数据库中删除的数据库角色的名称。

例如：删除前面建立的 teacher 角色：

```
sp_droprole  'teacher'
```

3. 添加角色成员

SQL Server 的系统存储过程 sp_addrolemember 用于给数据库角色添加成员，其语法是：

```
sp_addrolemember  '数据库角色名', '数据库用户名'
```

说明：

（1） “数据库角色名”是当前数据库中的数据库角色的名称。

（2） “数据库用户名”是添加到该角色的安全账户。“数据库用户名”可以是数据库用户、数据库角色、Windows 登录或 Windows 组。

例如：把前面建立的学生用户“1234120102”，添加到 student 角色中：

```
sp_addrolemember 'student','1234120102'
```

4. 取消角色成员

SQL Server 的系统存储过程 sp_droprolemember 用于取消数据库角色成员，其语法是：

```
sp_droprolemember  '数据库角色名', '数据库用户名'
```

其参数说明同系统存储过程 sp_addrolemember，这里不再赘述。

任务 3　为学生成绩管理系统数据库用户和角色赋予权限

一个登录用户，成为数据库用户后，必须具有相应的权限才能进行相关的操作，成绩管理系统中，给超级用户组或角色授予查看所有学生表，课程表和成绩表查询信息的权限。给所有的教师用户组或角色授予查看所有任课学生的成绩信息，插入、删除和修改学生成绩信息的权限。给所有的学生角色授予查看个人成绩信息的权限。下面分别从交互方式和指令两种方式介绍给用户或角色授权的方法。

一、交互方式给用户或角色授权

1.　交互方式给角色和用户授权

下面以给前面建立的超级用户角色 admin 授权为例，介绍以交互方式为用户或角色授权的过程。打开 Microsoft SQL Server Management Studio 的“对象资源管理器”→“数据库”→“ssms”→“安全性”→“角色”→“数据库角色”→“admin”，单击鼠标右键在弹出的快捷菜单中选择“属性”如图 4-58 所示，然后会弹出如图 4-59 所示的数据库角色属性窗口，单击其中的“检索”按钮会弹出如图 4-60 所示的添加对象窗口，选择其中的“特定类型的所有对象”单选按钮，然后单击“确定”按钮，会弹出如图 4-61 所示的选择对象类型窗口，选择其中“表”复选框，然后单击“确定”按钮，会又回到数据库角色属性窗口，不过这时已经添加了要设置权限的对象。在“安全对象”选项中选择要设置权限的对象，比如先设置 student 表的权限，在安全对象选项中选择 student，在对应下面的权限中选择授予“选择”权限如图 4-62 所示，以同样的方法，设置 admin 角色对 course 表和 score 表的权限。另外三个角色的权限设置这里不再赘述，由读者自己完成。

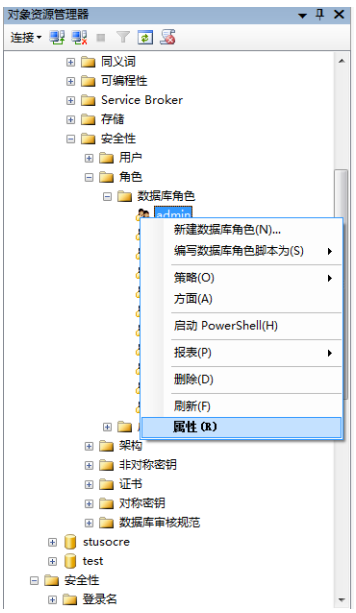


图 4-58　给角色授权

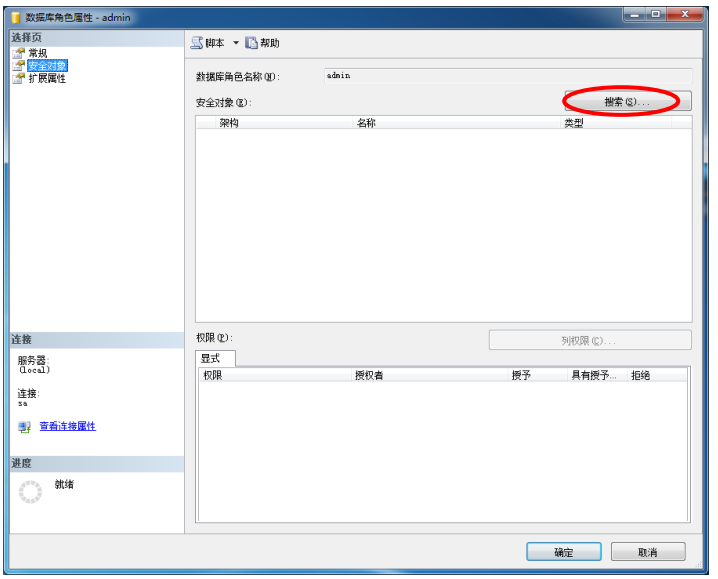


图 4-59　角色属性窗口

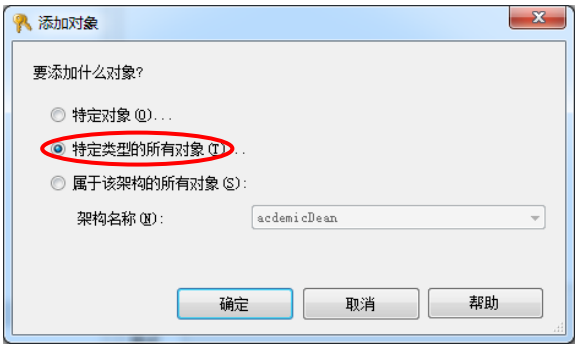


图 4-60　选择设置角色权限添加对象窗口

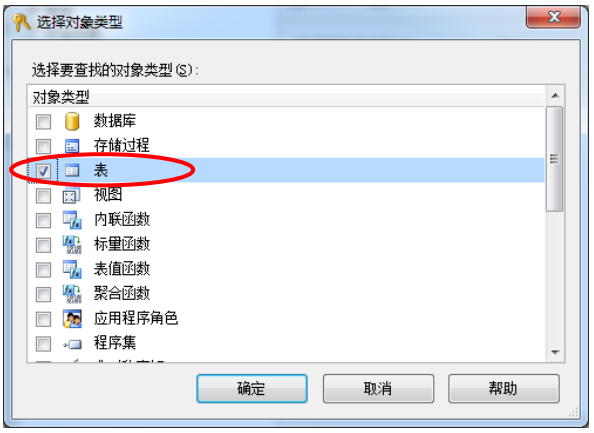


图 4-61　选择设置角色权限对象类型窗口

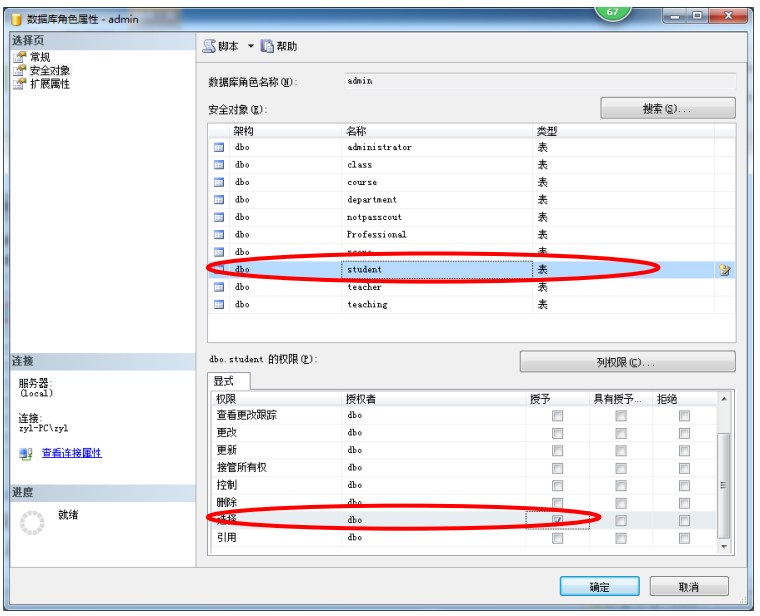


图 4-62　设置角色权限窗口

2. 交互方式撤销权限

交互方式撤销权限是在图中取消对应复选框中的选项即可。

3. 交互方式拒绝权限

交互方式拒绝权限是在图中取消对应复选框中的选项即可。

二、指令形式给用户或角色授权

1. 指令方式授予权限

授权通过 grant 指令实现，其语法是：

```
GRANT { ALL [ PRIVILEGES ] }  |权限[ ,...n ]
[ ON [ 表名或视图名 ][(<列名>)[ ,...n ]] | [<存储过程名>]
TO <数据库用户或角色> [ ,...n ]
[ WITH GRANT OPTION ]
```

说明：

1） ALL。

不推荐使用此选项，保留此选项仅用于向后兼容。它不会授予所有可能的权限。授予 ALL 参数相当于授予以下权限。

（1） 如果安全对象是数据库，则 ALL 对应 BACKUP DATABASE、BACKUP LOG、CREATE DATABASE、CREATE DEFAULT、CREATE FUNCTION、CREATE PROCEDURE、CREATE RULE、CREATE TABLE 和 CREATE VIEW。

（2） 如果安全对象是标量函数，则 ALL 对应 EXECUTE 和 REFERENCES。

（3） 如果安全对象是表值函数，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

（4） 如果安全对象是存储过程，则 ALL 表示 EXECUTE。

（5） 如果安全对象是表，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

（6）如果安全对象是视图，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

（7） PRIVILEGES。

包含此参数是为了符合 ISO 标准。请不要更改 ALL 的行为。

2） On 表示要授予其权限的对象，如果是表上授予权限，不提供具体的列名，则认为将影响该表的所有的列。

3） 指定表中将授予其权限的列的名称时。需要使用括号“()”。

4） To 表示将权限授予给定用户或角色。

5） With grant option 表示允许被授权者在获得指定权限的同时，还可以将指定权限授予其他数据库用户或角色。

为超级用户角色授权，具体实现代码如下：

```
grant  select on   student  to admin
grant  select on  course  to admin
grant  select on   score  to admin
```

2. 指令方式拒绝权限

拒绝权限通过 deny 指令实现，其语法是：

```
DENY { ALL [ PRIVILEGES ] }
| 权限 [ ,...n ]
[ ON [ 表名或视图名 ] [(<列名>)[ ,...n ]] | [<存储过程名>]
TO <数据库用户或角色> [ ,...n ] [ CASCADE]
```

说明：

（1） all、on 及 to 含义同 grant 指令。

（2） CASCADE 指示当前正在拒绝的权限也将从其他被该主体授权的主体中被拒绝。使用 CASCADE 参数时，还必须同时指定 GRANT OPTION FOR 参数。

例如，拒绝 student 角色对 score 表的 update 权限。

```
deny  update   ON  score  TO  student ;
```

数据库用户继承它所属角色的权限，并且可以通过交互方式和指令形式对权限进行扩展或拒绝。但只要用户继承的角色或用户本身授予的权限对某一权限是拒绝的，则该权限就是拒绝的。

3. 指令方式撤销权限

撤销权限通过 revoke 指令实现，其语法是：

```
REVOKE { ALL [ PRIVILEGES ] }  |权限[ ,...n ]
[ ON [ 表名或视图名 ][(<列名>)[ ,...n ]] | [<存储过程名>]
FROM <数据库用户或角色> [ ,...n ] [CASCADE]
```

参数说明：

1） ALL。

该选项并不撤销全部可能的权限。撤销 ALL 相当于撤销以下权限。

（1） 如果安全对象是数据库，则 ALL 对应 BACKUP DATABASE、BACKUP LOG、CREATE DATABASE、CREATE DEFAULT、CREATE FUNCTION、CREATE PROCEDURE、CREATE RULE、CREATE TABLE 和 CREATE VIEW。

（2） 如果安全对象是标量函数，则 ALL 对应 EXECUTE 和 REFERENCES。

（3） 如果安全对象是表值函数，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

（4） 如果安全对象是存储过程，则 ALL 表示 EXECUTE。

（5） 如果安全对象是表，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

（6） 如果安全对象是视图，则 ALL 对应 DELETE、INSERT、REFERENCES、SELECT 和 UPDATE。

2） ON 参数同 GRANT 指令。

3） FROM 要撤销权限的用户或角色。

4） CASCADE 指示当前正在撤销的权限也将从其他被该主体授权的主体中撤销。使用 CASCADE 参数时，还必须同时指定 GRANT OPTION FOR 参数。

例如，撤销超级用户 admin 对 course 表的 select 权限：

```
Revoke select from admin
```

4. 权限案例分析

```
sp addlogin  'sqluser', '123456';--创建登 SQL Server 录用户
use  ssms
go
sp grantdbaccess  'sqluser', ' sqluser ';
--把名为 sqluser 的登录用户变为 ssms 数据库中名 sqluser 的数据库用户
grant  select, insert, update  on  student  to  public ;
--给所有用户授权 student 表的 select、insert 和 update 权限
```

请分析经过前面的授权操作后，用 sqluser 账号登录，能否对 ssms 数据库中的 student 表进行 select 操作？

```
revoke  select  on  student  from  ' sqluser ';
--收回 sqluser 在 student 表上的 select 权限
```

请分析在收回 sqluser 登录用户在 student 表上的 select 权限后，再用 sqluser 账号登录，能否对 ssms 数据库中的 student 表进行 select 操作？

```
revoke select ,insert,update on student from public
--收回所有用户在 student 表上的 select、insert 和 update 权限
```

请分析在收回所有用户在 student 表上的 select、insert 和 update 权限后，用 sqluser 账号登录，能否对 student 表进行 select 操作？

```
grant  select, insert, update  on  student  to  public ;
--再次给所有用户授权 student 表的 select、insert 和 update 权限
deny  select   on  student  to  ' sqluser ' ;
--拒绝 sqluser 用户对 student 表的 select 权限
```

请分析在拒绝了 sqluser 用户对 student 表的 select 权限后，用 sqluser 账号登录，能否对 student 表进行 select 操作。

相关知识

一、数据库的安全性

数据库的安全性是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。安全性问题不是数据库系统所独有的，所有计算机系统都有这个问题。只是在数据库系统中大量数据集中存放，而且为许多最终用户直接共享，从而使安全性问题更为突出。系统安全保护措施是否有效是数据库系统的主要指标之一。数据库的安全性和计算机系统的安全性，包括操作系统、网络系统的安全性是紧密联系、相互支持的。实现数据库安全性控

制的常用方法和技术有：

（1） 用户标识和鉴别：该方法由系统提供一定的方式让用户标识自己的名字或身份。每次用户要求进入系统时，由系统进行核对，通过鉴定后才提供系统的使用权。

（2） 存取控制：通过用户权限定义和合法权检查确保只有合法权限的用户访问数据库，所有未被授权的人员无法存取数据。

（3） 视图机制：为不同的用户定义视图，通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

（4） 审计：建立审计日志，把用户对数据库的所有操作自动记录下来放入审计日志中，DBA 可以利用审计跟踪的信息，重现导致数据库现有状况的一系列事件，找出非法存取数据的人、时间和内容等。

（5） 数据加密：对存储和传输的数据进行加密处理，从而使得不知道解密算法的人无法获知数据的内容。

在一般的计算机系统中，安全措施是一级级层层设置的，用户要求进入计算机系统时，首先进行用户身份鉴定，只有合法用户才允许进入系统。对已进入系统的用户，DBMS 要进行存取控制，只允许用户执行合法操作。操作系统也会有自己的保护措施，但在此仅讨论与数据库有关的用户标识和鉴定、存取控制、视图和密码存储等安全技术。

二、SQL Server 身份验证

SQL Server 身份验证有三种形式：Windows 身份验证、SQL Server 身份验证和混合身份验证。

1. Windows 身份验证

Windows 身份验证使用 Windows 操作系统的安全机制验证用户身份，只有用户能够通过 Windows 用户账户验证，即可连接到 SQL Server 的服务器，适用于仅在组织内部访问，在使用 Windows 身份验证时，SQL Server 检测当前使用的 Windows 登录账号，并在系统注册表中查找该用户，以确定该用户是否有权登录。

2. SQL Server 身份验证

SQL Server 身份验证是通过登录账号和密码的匹配进行验证的，因此采用 SQL Server 身份验证时，用户在连接 SQL Server 时必须提供登录账号和登录密码。用户的登录账号和密码信息存储在系统表 syslogins 中，如果登录用户输入的登录信息与系统表 syslogins 中的某条记录相匹配时则登录成功。

3. 混合验证

如果用户在登录时提供了 SQL Server 登录账号，则系统将使用 SQL Server 身份验证对其进行验证。如果没有提供 SQL Server 登录账号或请求 Windows 身份验证，则使用 Windows 身份验证对其进行身份验证。

与 SQL Server 身份验证相比，Windows 身份验证有某些优点，主要是由于它与 Windows 安全系统的集成。Windows NT 安全系统提供更多的功能，如安全验证和密码加密、审核、密码过期、最短密码长度，以及在多次登录请求无效后锁定账户。

这些模式在安装过程中指定或使用 SQL Server 对象资源管理器指定。

通过SQL Server对象资源管理器修改SQL Server的身份验证模式，具体操作步骤如下：在 SQL Server Management Studio 的对象资源管理器中，右键单击“服务器实例”，在弹出的快捷菜单中选择“属性”菜单项，如图 4-63 所示。会弹出如图 4-64 所示的服务器属性窗口，在服务器属性窗口中选择“安全性”，在“服务器身份验证”选项中，选择要设置的身份验证模式，然后单击“确定”按钮，完成了身份验证模式的设置。

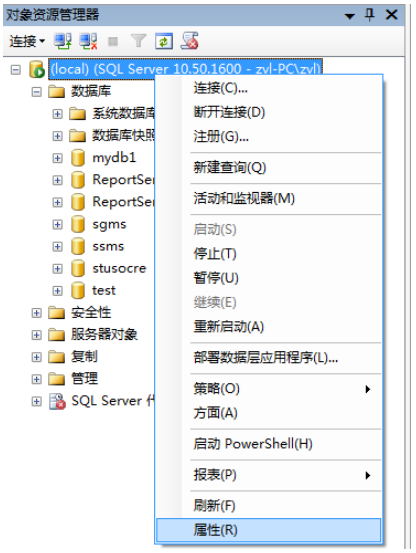


图 4-63　选择数据库服务器属性窗口

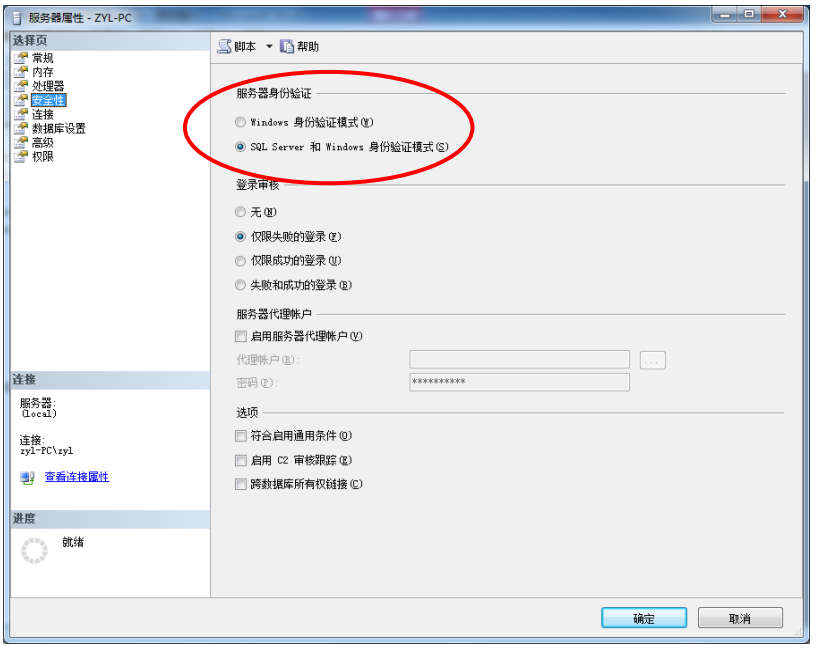


图 4-64　修改数据库身份认证模式

习题

1. 简述 SQL Server 身份认证模式？

2. 简述 SQL Server 的角色有哪几种？

3. 简述 SQL Server 安全控制机制。

4. 简述 SQL Server 的登录用户和数据库用户之间及数据库用户与数据库角色之间的关系。

5. 简述架构的概念。

实训项目

1. 请为学生成绩管理系统创建四种角色，admin、teacher、acdemicDean、student，并授予其相应权限。给 admin 角色授予查看 student 表，course 表和 score 表查询信息的权限。给 teacher 角色授予查看所有任课学生的成绩信息，插入、删除和修改学生成绩信息的权限。给 student 学生角色授予查看个人成绩信息的权限，给 acdemicDean 授予查看、插入、删除和修改 course 表及 student 表的权限。并为每个角色创建至少两个用户。用创建的登录用户验证权限设置的有效性。

2. 假设要对全校学生开放对 ssms 数据库中学生选课成绩的查询权限，但禁止他们对选课成绩的修改，而且一个院系的学生只能查看本系同学的选课成绩，不能查看其他系同学的选课成绩，但允许全校学生在数据库 ssms 的复查成绩申请表 GradeFC（Dept，Sno，Cno，Grade，Description）中填写复查成绩申请记录，请给出一种安全管理方案，并通过实验验证你的方案。